Escuela Politécnica Superior

Departamento de Tecnología Electrónica y de las Comunicaciones

ADVANCES IN SEMANTIC-GUIDED AND FEEDBACK-BASED APPROACHES FOR VIDEO ANALYSIS

PhD Thesis written by
Juan Carlos San Miguel Avedillo
under the supervision of
Prof. José María Martínez Sánchez

Madrid, September 2011

Department: Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid, Spain

PhD Thesis: Advances in semantic-guided and feedback-based approaches for video analysis

Author: **Juan Carlos San Miguel Avedillo**
Ingeniero de Telecomunicación
(Universidad Autónoma de Madrid)

Supervisor: **Jose María Martínez Sánchez**
Doctor Ingeniero de Telecomunicación
(Universidad Politécnica de Madrid)
Universidad Autónoma de Madrid , Spain

Year: 2011

Comittee: President: **Narciso García Santos**
Universidad Politécnica de Madrid, Spain

Secretary: **Jesús Bescós Cano**
Universidad Autónoma de Madrid, Spain

Vocal 1: **Carlo Regazzoni**
Università degli Studi di Genova, Italy

Vocal 2: **Andrea Cavallaro**
Queen Mary University of London, UK

Vocal 3: **François Bremond**
INRIA, France

The work described in this Thesis was carried out within the Video Processing and Understanding Lab at the Dept. of Tecnología Electrónica y de las Comunicaciones, Escuela Politécnica Superior, Universidad Autónoma de Madrid (from 2008 to 2011). It was partially supported by the Consejería de Educación of the Comunidad de Madrid and by the European Social Fund through a Contrato de Personal Investigador de Apoyo, by the Spanish Government (TEC2007-65400-SemanticVideo), by the Spanish Administration agency CDTI (CENIT-VISION 2007-1007), by the Comunidad de Madrid (S-0505/TIC/0223-ProMultiDis-CM) and by Cátedra Infoglobal-UAM for "Nuevas Tecnologías de video aplicadas a la seguridad".

Part of the work related to video object tracking evaluation was done while visiting the Multimedia & Vision Research Group at the Queen Mary University of London (UK) under the supervision of Prof. Andrea Cavallaro from May 24 to August 24, 2009 and from May 26 to August 26, 2010.

To Sonsoles and my family.

Success usually comes to those who are too busy to be looking for it.
- Henry David Thoreau (1817-1862)

Acknowledgments

This thesis is the result of four years of work. There has been ups and downs but I've enjoyed every moment. It is a pleasure to thank the many people who made this thesis possible.

First I would like to express my gratitude to my supervisor, Dr. José María Martínez Sánchez, for his helpful discussions and thorough editing of this thesis. I really appreciate his constant support and advice since I joined the VPU-Lab. I also thank him for believing in me and letting me to pursue the research directions I was interested in within the topic of my thesis.

Dr. Andrea Cavallaro also deserves special thanks for his sincere advice and his guidance during numerous occasions. While I was visiting the Queen Mary University of London in 2009 and 2010, I learned many things from several enriching discussions with him.

I would also thank Dr. Jesús Bescós Cano for his advice and suggestions during the past years. We have shared many exciting projects that have allowed me to learn from his experience and knowledge.

I am indebted to Sonsoles for her love, support and patience that made everything easier for me. She has been close every moment. She deserves much more than just a 'thank you'. Not only now, but also in the future years.

I also wish to thank all members of VPU-Lab for their help and friendship. Each one has contributed to finish this work in some way. We shared many unforgettable moments that made my PhD experience very exciting. I am specially grateful to Miguel Angel, Javi, Luis I, Luis II, Fabrizio, Víctor Fernández, Víctor Valdés, Fernando, Álvaro I, Álvaro II, Álvaro III, Marcos and Virginia. Additionally, during the PhD period I got the opportunity to supervise some undergraduate students from whom I shaped my working attitudes. Thank you for trusting me.

Last but not least, this work would not have been possible without the continued support provided by family and specially my mother.

Juan Carlos San Miguel Avedillo
October 2011

Abstract

Nowadays, the huge amount of available video content demands the creation of automatic systems for its understanding. In particular, human event recognition has become a relevant research area motivated by the variety of promising applications in the private and public sectors. In this context, system design is a challenging task as many issues arise related with the structure and performance in specific scenarios. For improving current systems, the *Cognitive Computer Vision* paradigm was recently proposed to study the relation of the system with its environment, its results and the available resources. However, its use in video analysis presents limited success in real scenarios. This thesis addresses the use of *semantics* (high-level knowledge representations) and *feedback* processing schemes for human event recognition in video content.

The first part starts by modeling the high-level knowledge related to video events in terms of the application domain and the analysis system. This model suits the needs of many applications being not restricted to any specific implementation. Then, its practical use for guiding video analysis is explored in two situations: automatic workflow composition and human-related event recognition. The former is focused on the automatic selection and ordering of the most appropriate tools among the available ones in the system for the analysis of a specific domain. The latter is concentrated on defining adequate structures for video event recognition considering temporal information and the uncertainty of the low-level analysis. Experiments on real datasets demonstrate the efficiency of the two described practical use cases.

In the second part, a generic feedback processing scheme is proposed to allow a variable analysis effort according to the input data complexity and the output quality estimation. Then, it is applied to the processing stages of a traditional video event recognition system. Compared to the traditional (sequential) system, the use of feedback increases the precision and highly reduces the computational cost. Later on, this thesis focuses on a critical feedback-related issue: output quality estimation without ground-truth data. The foreground segmentation and object tracking stages are studied by providing taxonomies for current literature and by comparing the most representative approaches. Results show that different approaches should be used to detect specific performance characteristics. Finally, a novel approach is presented for object tracking evaluation without ground-truth in which its adaptive capability, that bounds the computational cost, makes it usable for long sequences where the tracking algorithm is expected to fail.

Resumen

Actualmente, la gran cantidad de contenido visual demanda la creación de herramientas automáticas de análisis. En particular, la detección de eventos (actividades) humanas se ha convertido en un área de investigación muy activa motivado por la variedad de aplicaciones en el sector privado y público. En este contexto, el diseño de sistemas complejos es una tarea muy difícil debido a cuestiones relacionadas con la estructura y el rendimiento en distintos escenarios. Para mejorar los sistemas actuales, el paradigma de *Visión Cognitiva por Computador* fue recientemente propuesto para estudiar la relación del sistema con su entorno, sus resultados y los recursos existentes. Su aplicación al reconocimiento de eventos es limitada en escenarios reales. Esta tesis trata del uso de representaciones *semánticas* del contenido y esquemas de procesado *realimentado* para el análisis de eventos en vídeo.

En la primera parte, se comienza modelando el conocimiento de alto nivel relacionado con el dominio de aplicación y el sistema de análisis. Este modelo que satisface las necesidades de una gran variedad de aplicaciones sin estar restringido a una implementación específica. Después, se explora su uso práctico en la composición automática del flujo de trabajo y el reconocimiento de eventos. La primera está centrada en la selección y ordenación automática de las herramientas más apropiadas para el análisis de un dominio de aplicación. La segunda define estructuras adecuadas para el reconocimiento de eventos considerando la temporalidad y la incertidumbre del análisis. Los experimentos con datos reales demuestran la eficiencia de ambas propuestas.

En una segunda parte, se propone un esquema genérico de realimentación que permite variar el esfuerzo de análisis acorde con la complejidad de los datos de entrada y la calidad de los resultados. Después, éste se aplica a las etapas de procesamiento de un sistema tradicional de reconocimiento de eventos. Comparado con el sistema tradicional (secuencial), se incrementa la precisión del sistema y se reduce drásticamente el coste computacional. Más adelante, se analiza un aspecto crítico de la realimentación: la evaluación del resultado sin información etiquetada. La etapa de segmentación de frente y seguimiento de objetos son estudiadas mediante la propuesta de taxonomías para las aproximaciones actuales y la comparación de las más representativas. Finalmente, se presenta un algoritmo novedoso para la evaluación del seguimiento de objetos sin datos etiquetados en el cual su capacidad adaptativa, que limita su coste computacional, lo hace aplicable en largas secuencias donde el algoritmo de seguimiento se espera que falle.

Contents

List of Figures

List of Tables

Part I

Introduction

Chapter 1

Introduction

1.1 Motivation

Since the middle of the 20^{th} century, image capture devices have been experimenting an enormous technological evolution whereas decreasing their production costs. The recent digitization of capture devices and the high increase of computational power have attracted the attention of the private and public sectors to massively deploy vision-based systems. Hence, the amount of video content created by such systems has dramatically grown during last years. The optimal use of these data in the shortest possible time has become a critical issue.

This situation has created a challenge into the research community to devise algorithmic approaches for automatic understanding of the video content. This task consists of extracting descriptors to represent the semantic concepts that humans perceive when observing a video sequence. A wide range of applications come from this research such as medical imaging, machine inspection, biometrics, surveillance and content-based indexing. Moreover, complications appear when developing complex systems as the human operator is needed for many design tasks such as the system structure, the selection of the appropriate algorithms and the performance in different scenarios. Thus, this design task requires skilled expertise in many different areas.

For improving the capabilities of vision-based systems, the *Cognitive Computer Vision* paradigm (CCV) was recently proposed [Christensen and Nagel, 2006]. It pursues to achieve more robust and adaptable systems through further developing the relationship between the system and its environment. The human cognition process is emulated to recognize structures and learn from the experience instead of providing a simple reaction to inputs (such as traditional vision) [Brachman, 2002]. This paradigm relies on two key elements: visual learning and task-based attentional control. The former refers to the representation, learning and recognition of objects of interest. The latter regards the interaction of the vision system with its environment, available resources

and the perceived tasks. The EU projects CogVis[1], CogViSys[2] and SCOVIS[3], among others, demonstrate the research interest on this area.

The current application of the CCV paradigm to video understanding is in its early stages and the aim of giving *cognition* is still far from being successful. One of the major hurdles is the processing of unexpected data. Recently, few CCV-based approaches have been proposed to deal with this issue for learning the temporal structure of vision tasks, recognizing complex objects and spatio-temporal reasoning for scene interpretation [Vernon, 2008]. However, the objective of creating a general-purpose system with human-like capabilities remains unsolved.

During the last years, the automatic recognition of human-related events has become a relevant research area motivated by the variety of promising applications in many domains such as video surveillance, smart environments and sports analysis. It involves the detection of ongoing human-executed activities in video. The challenges involved in estimating motion and recognizing people from videos have limited the success of existing approaches. The high interest is exhibited by the notable amount of research projects including ADVISOR[4], ICONS[5], CAVIAR[6], AVITRACK[7], ETISEO[8], CARETAKER[9], BEWARE[10], SAMURAI[11] and ViCoMo[12].

In the context of human event recognition, a proper high-level semantic representation (e.g., knowledge related to objects and their behavior) is required as there is a broad spectrum of application domains (e.g., traffic monitoring, smart meeting, indoor surveillance). Besides, this representation is essential to facilitate the interoperability and integration between systems. Currently, there are few approaches that provide a detailed description of such domain-related semantics (e.g., [Francois et al., 2005; Fernandez, 2010]). However, they do not suggest the right algorithms to extract the semantic information and, therefore, an automatic solution to the analysis problem is not provided. An integrated representation of domain and system knowledge would be desirable to model the interaction of the system with its environment and capabilities (e.g., for selecting the most appropriate recognition algorithms).

Furthermore, most of the existing literature for learning and recognizing events consider as valid the results obtained from previous processing stages such as object segmentation, tracking and recognition, which are traditionally supposed to be independent among them. Nevertheless, the remarkable relation that the results present with the application domain and the close

[1] IST-29375, 2001-2004 (available at http://www.comp.leeds.ac.uk/vision/cogvis/).
[2] IST-29404, 2001-2004 (available at http://cogvisys.iaks.uni-karlsruhe.de/).
[3] IST-216465, 2007–2013 (available at http://www.scovis.eu/)
[4] IST-11287, 2000–2002 (available at http://www-sop.inria.fr/orion/ADVISOR/).
[5] DTI/EPSRC LINK, 2001–2003 (available at http://www.eecs.qmul.ac.uk/~sgg/ICONS/).
[6] IST-37540, 2002-2005 (available at http://homepages.inf.ed.ac.uk/rbf/CAVIAR/).
[7] AST-CT-502818, 2004–2006 (available at http://www.avitrack.net).
[8] Techno-Vision, 2005–2007 (available at http://www-sop.inria.fr/orion/ETISEO).
[9] IST-027231, 2006-2008 (available at http://cordis.europa.eu/ist/kct/caretaker_synopsis.htm).
[10] EP/E028594/1, 2007–2010 (available at http://www.eecs.qmul.ac.uk/~sgg/BEWARE/).
[11] IST-217899, 2008–2011 (available at http://www.samurai-eu.org/).
[12] ITEA2-08009, 2009–2012 (available at http://www.vicomo.org/).

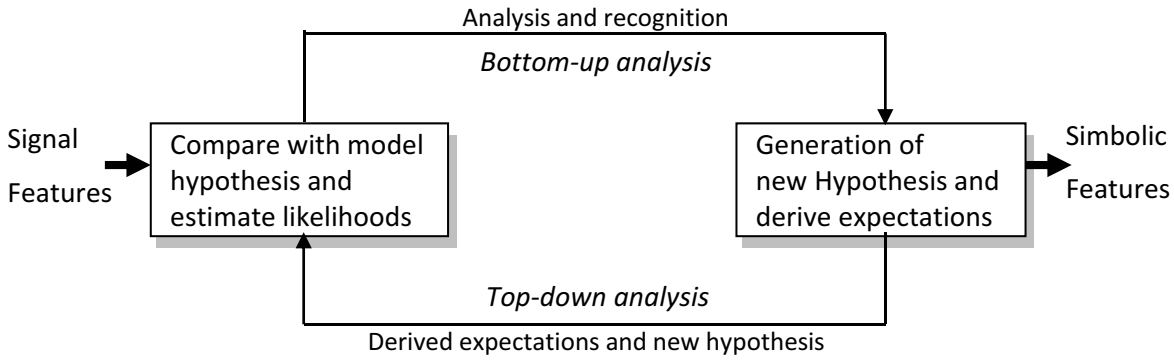


Fig. 1.1. The iterative cycle for hypothesis generation and verification (from [Town, 2006]).

dependence of the algorithm results with respect to other subsequent algorithms of the analysis chain (e.g., a poor segmentation makes recognition very difficult) are well-known shortcomings [Izquierdo et al., 2004]. It is accepted that most of the basic stages of video event recognition (segmentation, tracking and recognition) are highly related. For example, segmentation takes into account motion continuity between consecutive frames and tracking takes into account some criteria about connectivity, compactness, aspect ratio, etc. The feedback of information between the processing stages of a system can be used to improve the overall performance (as reported for the traffic monitoring domain by [Nagel, 2004]).

Under the above-mentioned conditions, the improvement of current human event recognition systems can be achieved by adapting the main principles of the CCV paradigm to video analysis: the representation of high-level semantics and the use of a feedback path for modeling the interaction between the processing stages of a system, verifying the analysis hypothesis (e.g., the recognized events) and adapting the video analysis processes to the current environment.

1.2 Objectives

The main objective of this thesis is *to explore the application of semantic representations and feedback processing schemes to the recognition of human-related video events*[13]. Inspired by the analysis cycle of [Town, 2006] (see Fig. 1.1), we propose to understand the video data in a dynamic context as an iterative process of generation and validation of hypotheses. Thus, bottom-up analysis creates hypotheses from raw data (e.g., recognized events), and afterwards, top-down analysis confirms or rejects them and provides additional hypotheses (feedback path).

For achieving this objective, we propose to study the following areas:

[13] An event in the general sense is defined as something that happens at a given place and time. Among existing literature, several terms have been used for referring to the recognition of human-related events such as action, activity, scenario or behavior. In this document, we will consider the term event for describing occurrences of interest in video sequences that are executed by humans.

- Semantic representation. Hierarchical descriptions are studied for providing an integrated representation of domain knowledge (e.g., the observable entities such as objects and events) and system knowledge (e.g., available analysis tools such as foreground segmentation and tracking). Additionally, relations between both knowledge sources are explored.

- Semantic-guided video analysis (development of the bottom-up path). We investigate the use of the hierarchical descriptions for developing self-configurable video analysis frameworks and for structured recognition of video events considering the uncertainty of the involved processing stages and the temporal information of events.

- Feedback-based video analysis (development of the top-down path). We propose to define a generic feedback model for video analysis. Therefore, techniques for generating new hypothesis (e.g., new foreground segmentation masks) and for estimating their performance (e.g., quality of segmentation masks) have to be studied. Finally, we explore its use for improving the robustness of a complete video-surveillance system.

- Quality estimation without ground-truth for foreground segmentation and object tracking. For the proposed feedback analysis model, a key issue is the quality estimation of the analysis results without using annotated data that are very expensive to produce and usually cover a small portion of data variability besides not being applicable to control the system performance in real scenarios. Thus, we explore this evaluation for two widely used stages in video event recognition: foreground segmentation and object tracking.

1.3 Major contributions

The significant novel contributions of this thesis are summarized below:

1. Hierarchical representation of the knowledge related to the application domain (referring to events), to the system that analyzes the video content and to their relations.

2. A scalable and distributed framework for video analysis based on event-related semantics. Its main advantage is the capability to select the appropriate algorithms from available ones for automatic workflow composition to analyze different domains.

3. A structured event recognition approach guided by the event-related semantics that considers the uncertainty of the involved analysis stages.

4. Definition of a model for feedback-based analysis. It allows to adjust analysis effort guided by the input data complexity and the output quality estimation. Moreover, state-of-art techniques are suggested to be used in the proposed model.

5. Application of the feedback-based analysis model to the processing stages of an event recognition system for video surveillance. Coordination of the applied feedback strategies to improve system performance in terms of accuracy and computational cost.

6. Taxonomies for performance evaluation methods (i.e., output quality estimators) without ground-truth information for foreground segmentation and object tracking.

7. Comparative evaluation of representative approaches for output quality estimation without ground-truth information for foreground segmentation and object tracking.

8. Method for track quality estimation without ground-truth. This method is based on identifying the successful operation of the tracker and on applying an additional tracker in reverse direction to estimate the quality of the tracking result.

1.4 Structure of the document

This document is structured in five parts, which are organized as follows:

- Part I: Introduction
  - *Chapter 1: Introduction.* This chapter presents the motivation, the objectives, the main contributions and the structure of this thesis.
- Part II: Semantic-guided video analysis
  - *Chapter 2: Representation of video event semantics.* It describes the hierarchical framework for representing the semantics related to video events.
  - *Chapter 3: A self-configurable framework for video analysis.* It proposes a scalable and distributed framework for video sequence analysis that automatically estimates optimal workflows based on the proposed semantic representation.
  - *Chapter 4: Recognition of single-view human-related video events.* It proposes a structured recognition of video events based on the proposed semantic representation.
- Part III: Feedback-based video analysis
  - *Chapter 5: Feedback-based analysis model.* It proposes a generic model for feedback-based analysis inspired by classical closed-loop control systems.
  - *Chapter 6: Feedback-based event recognition.* It proposes the application of the feedback model to an event recognition system for video surveillance.

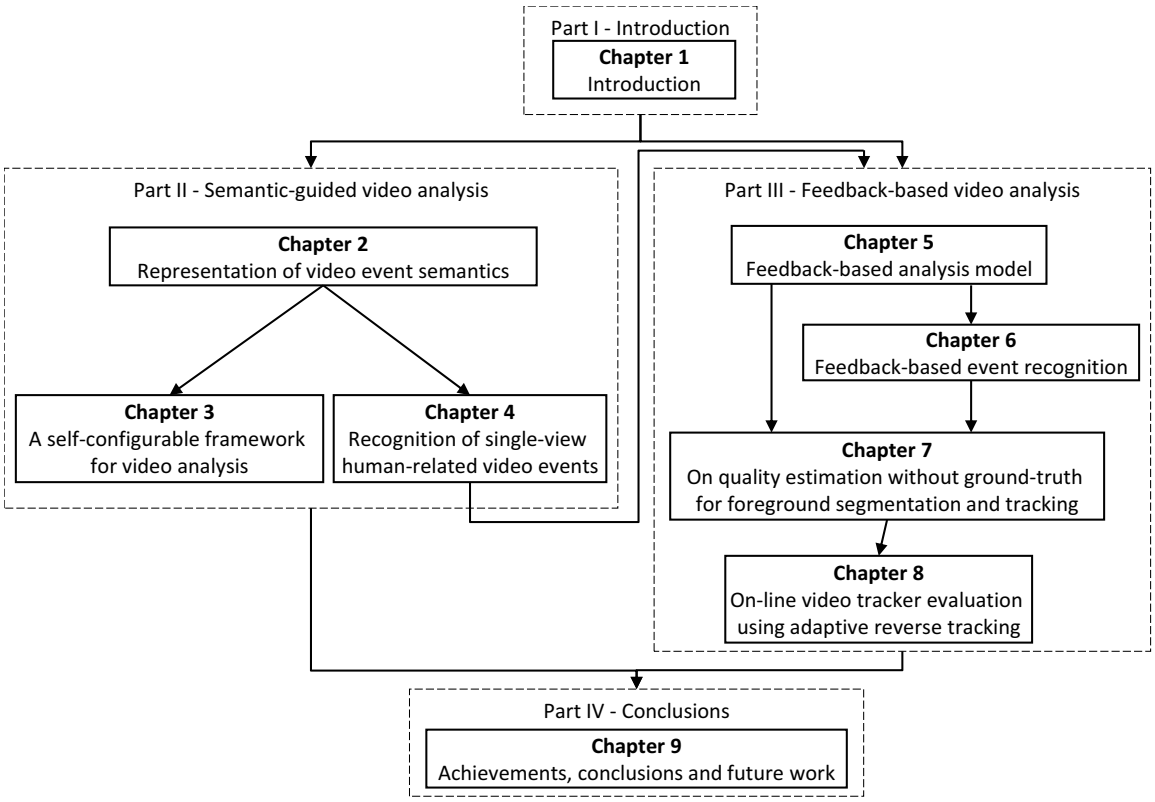


Fig. 1.2. Dependence among the chapters of this thesis.

  - *Chapter 7: On quality estimation without ground-truth for foreground segmentation and tracking.* It describes a comparative study of existing approaches for estimating the output quality (e.g., performance) without ground-truth information.
  - *Chapter 8: On-line video tracker evaluation using adaptive reverse tracking.* It proposes a novel method for performance evaluation of video trackers and demonstrates its application for the particle filter framework.

- Part IV: Conclusions
  - *Chapter 9: Achievements, conclusions and future work.* It concludes this document summarizing the main results and future work for its extension.

- Part V: Appendixes
  - *Appendix A: Publications.*

The relationships between chapters and parts of the thesis are depicted in Fig. 1.2.

Part II

Semantic-guided video analysis

Chapter 2

Representation of video event semantics

2.1 Introduction[1]

The active research carried out in the area of event recognition and the proliferation of vision-based systems in a wide variety of domains (such as banks, airports and stores) have created the demand to design high-level descriptions for optimally accessing and analyzing the content. Recently, ontology-based approaches have gained interest as they standardize event definitions allowing easy portability and interoperability between different systems. We propose an ontology-based approach to represent the information related to events in terms of the application domain, the analysis system and the user of the analysis results.

In this chapter, we will firstly present the related work for the semantic representation of events in section 2.2. Then, we overview the proposed ontology-based representation in section 2.3. After that, sections 2.4, 2.5 and 2.6 define the entities to describe the domain, the system and the user knowledge. Finally, section 2.7 summarizes the chapter with some conclusions.

2.2 Related work

2.2.1 Knowledge learning and representation

In [Kompatsiaris and Hobson, 2008], two types of approaches can be identified for representing and acquiring knowledge: *implicit* and *explicit*. The former relies on the use of machine learning techniques and therefore, the relations between low-level data and high-level semantics are inherently learned. The latter includes the approaches that describe prior knowledge with predefined facts, models and rules. Thus, a formal definition of the semantics is provided for each specific context by an expert.

[1] This chapter is an extended version of the publication *"J.C. SanMiguel, J.M. Martínez, A. García. An ontology for event detection and its application in surveillance video. Proc. of the IEEE Int. Conf. on Advanced Video and Signal based Surveillance, pp. 220-225. Genoa (Italy), Sept. 2009".*

Implicit	Explicit
Advantages	Advantages
• Models automatically learned. Experts provide training samples. • High precision within a domain. • Uncertainty of low-level analysis is implicitly considered	• Compact representation of domain knowledge. • Allows representation of complex semantics. • Facilitates the interoperability between systems.
Disadvantages	Disadvantages
• No clear number of needed examples for optimal modeling. • Complex temporal models are very hard to learn. • Re-training is usually required for the analysis of new domains.	• Content formally described, refined, adapted and integrated by domain experts. • In most cases, the right algorithm to use is not suggested. • Uncertainty of low-level analysis difficult to be included in the model.

Table 2.1: Comparison between the main approaches for knowledge learning and representation.

As recently pointed by several authors [Fernandez, 2010; Lavee et al., 2010b], each approach presents different characteristics and its use depends on the kind of knowledge to be modeled. Implicit approaches allow an easy management of high dimensional problems and do not require detailed domain modeling. However, their application to new domains is limited and they are not able to model complex semantics (e.g., events involving several actors) over long time periods [Lavee et al., 2009]. On the other hand, explicit approaches propose a formal representation based on human-expertise. This representation is more compact than the one of the implicit approaches but the complexity of the modeling task is higher. Moreover, explicit approaches allow an easy use of the existing reasoning methods to infer new types of knowledge and interoperability between systems. A summary of their advantages and disadvantages is provided in Table 2.1.

In the rest of this section, we review the literature related with the explicit representation of high-level semantics. The implicit representation approaches have been excluded as they are out of the scope of the work presented in this chapter.

2.2.2 Explicit representation of high-level semantics related to video events

As commonly agreed in recent surveys for event detection [Turaga et al., 2008; Lavee et al., 2009; Ballan et al., 2011; Aggarwal and Ryoo, 2011], explicit representation of video events can be categorized into *syntactic*, *logic* and *ontology* based approaches.

Syntactic-based approaches rely on grammar models that express the event structure with a set of production rules. They specify how sentences (events) can be constructed using a string of symbols or words (sub-events). This representation easily describes sequential combination of sub-events. For example, Context-Free Grammar (CFG) [Bobick and Wilson, 1997] and Stochastic Context-Free Grammar (SCFG) [Ivanov and Bobick, 2000] have been used for event modeling in video analysis. Recently, [Ryoo and Aggarwal, 2009] extended this formalism for defining temporal combinations of events by using Allen's predicates [Allen, 1994] that define temporal relations between symbols (e.g., events). However, other complex relations among events are very hard to model (e.g., concurrency). Besides, they are limited in their use for domains that share a structured knowledge base and for defining contextual information.

Logic-based approaches model each event using formal logical rules. For example, [Medioni et al., 2001] proposed a representation to recognize single-agent events based on a temporal combination of short-term events that are recognized using a set of rules applied on moving object properties. Moreover, [Shet et al., 2005] defined a model that uses logic programming to describe events as logical rules between sub-events expressed using the Prolog programming language. In a similar way, [Artikis et al., 2010] described how events can be recognized using logic programming based on a set of first-order logic, including temporal formalism, for representing and reasoning about events and their effects. Similar to the syntactic category, logic-based approaches provide an efficient way to incorporate domain knowledge but they share the same disadvantages for defining complex events and using contextual information. Hence, a standard, extensible and shareable representation is needed for defining complex high-level semantics.

Although ontology-based knowledge description has been widely used in the field of Artificial Intelligence, only recently it has been proposed to formalize the information related to video events. It is based on creating a hierarchical description of the spatio-temporal event structure that offers a great advantage for semantic annotation [Ballan et al., 2011], for training recognition models [Town, 2004] and for defining the relations with extraction tools [Dasiopoulou et al., 2005]. However, most of the existing approaches propose ad-hoc ontologies targeted towards a specific domain. Despite their accuracy, they are limited for knowledge sharing and the analysis of similar domains. For example, [Hakeem and Shah, 2004a] defined the meeting domain with specific entities for objects (e.g., hands and heads) and events (e.g., event, behavior and genre). Hence, the type of content to be analyzed is highly restricted. Similar models have been proposed for indoor surveillance [Town, 2004], nursing home [Chen et al., 2004] and soccer [Ballan et al., 2010] domains. Thus, a generic hierarchical description is needed to solve these drawbacks.

Generic ontology-based descriptions define a hierarchical structure for the analysis problem (e.g., event detection) identifying its key concepts (e.g., objects and events) and providing an extension with the prior knowledge of each modeled domain (e.g., underground monitoring). For instance, [Nagel, 2004] proposed a hierarchical ontology structured in terms of events, verbs, episodes, stories, etc, and its extension for the traffic monitoring domain. Most of the latest approaches are based on Allen's predicates [Allen, 1994]. In particular, [Vu et al., 2003] represented events by specifying necessary conditions using Allen's temporal predicates in a conjunctive way (i.e. only 'and' concatenations are allowed, not 'or'). Moreover, [Georis et al., 2004] extended this previous work for bank scenarios by proposing a formalization of the available knowledge into three entities: physical objects (real world objects present in the scene), components (lists of states and events) and constraints (relations between physical objects, events or sub-events). An extension of these previous works is presented in [Fusier et al., 2007] to address long-term complex event description by defining an event hierarchy composed of primitive states (visual properties), composite states (combination of primitive states), primitive events (change of values for a primitive state) and composite events (combination of primitive states/events). Additionally, an object hierarchy is proposed for moving and static objects of the scene that allows the definition of complex relations involving objects of different types (e.g. individuals and vehicles). Similarly, [Hongeng et al., 2004] proposed an event hierarchy based on the composition of several action threads defined by the temporal concatenation (simple and complex) and the number of actors (single and multiple). However, no object hierarchy is given and the events models are based only on changes of the properties of the scene moving objects. Additionally, their hierarchy of events is limited to three levels. The successful application of this proposal has been demonstrated for the parking lot [Hongeng et al., 2004] and bank monitoring [Akdemir et al., 2008] domains. More recently, [Fernandez et al., 2008] introduced a deterministic formalism to hierarchically define events (as the change of object properties in a specific context or location), the observable objects and the relations between them. As previous approaches, its main drawback is the depth of the event hierarchy (limited to three levels). Besides, it is limited for representing concurrency and non-sequential temporal relations that are not straightforward to model. Furthermore, [Martinez-Tomas et al., 2008] defined a hierarchical ontology for the existing objects in the scene, the events and the purpose of the analysis (e.g. monitoring). Events are categorized attending to its composition (primitive and composed) and transient nature (instantaneous and fluent). However, the possible relations between events and objects (temporal, logical and spatial) are not clearly defined and, therefore, the extension of the model for describing complex events is difficult.

Furthermore, some ontology-based approaches integrate a simple description of system capabilities (e.g., semantic extraction tools) within the knowledge description. For example, [Dasiopoulou et al., 2005] connected object properties and detection algorithms in a common know-

ledge base. Then, the right association between algorithms and objects is determined through a set of rules formulated in F-logic. However, this proposal is tailored towards object detection and its extension to event recognition is not straightforward. Similarly, [Bai et al., 2007] defined a soccer ontology that links the events, objects and algorithms of this domain. Description Logic is used to select the appropriate algorithms to apply by exploiting their relationship with objects. However, it does not provide hierarchical descriptions and, therefore, the modeling of complex semantics is difficult. Moreover, [Martinez-Tomas et al., 2008] proposed to include a basic hierarchy of the structures employed for recognizing the modeled entities. Despite the effort done, a proper description of the system capabilities has not been found in the current literature.

As a result of the huge amount of existing ontology-based descriptions, some formal languages have been proposed to formalize ontological event definitions such as the CASE^E [Hakeem et al., 2004b] and the VERL [Francois et al., 2005] ones. The CASE^E language uses the concept of *case frame* to define the semantic of the occurrences in a video sequence. Different *case frames* are composed to define simple semantics. It allows to include event hierarchical descriptions and support causal relationships among objects. Based on Allen's logic, the VERL language defines the temporal and logical relationships between the video entities. Event hierarchy distinguishes between single-thread (linear event combination) and multi-thread (concurrent event combination). However, it does not suggest an implementation to recognize the events. More recently, extensions of the VERL language have been proposed for complex interactions for encoding static knowledge using the SWRL logic language [Snidaro et al., 2007a] and for managing domain context [Snidaro et al., 2009]. Furthermore, [Vezzani and Cucchiara, 2010] has contributed to knowledge sharing by proposing a formal language for annotating, retrieving and distributing metadata extracted from the analysis of video surveillance sequences.

2.2.3 Conclusion

The syntactic and logic based approaches offer a quick way of incorporating domain knowledge into applications. However, their extensibility among different, albeit related, domains is limited. On the other hand, ontology-based approaches propose a structured and accurate high-level description of the video content. Most of the existing approaches define a common base of knowledge that is extended for each application domain. Nevertheless, there is no detailed description of the visual agents involved in the recognition of the semantic concepts that can be considered crucial for the automatic deployment and update of analysis systems.

In this situation, we propose an ontology-based approach to hierarchically represent the information related to events and their extraction tools. This description defines the knowledge of the application domain, the analysis system and the user of the system results. In Table 2.2 we compare the main reviewed approaches with our proposal in terms of aspects modeled at the object, event and system levels.

Reference	Object modeling		Event modeling				System modeling	
	Hierarchy	Spatial	Hierarchy	Event relations			Object	Event
	levels	relations	levels	Context	Temporal	Logical	analysis	analysis
Syntactic								
[Ivanov and Bobick, 2000]	1	NA	2	NA	S	S	NA	NA
[Ryoo and Aggarwal, 2009]	1	C	U	NA	C	C	NA	NA
Logic								
[Medioni et al., 2001]	1	NA	2	NA	C	S	NA	NA
Ontology								
[Hongeng et al., 2004]	1	S	3	NA	C	C	NA	NA
[Francois et al., 2005]	4	S	3	S	S	C	NA	NA
[Fusier et al., 2007]	2	S	3	S	C	C	NA	NA
[Bai et al., 2007]	1	NA	1	NA	NA	S	NA	S
[Martinez-Tomas et al., 2008]	4	S	U	S	NA	S	S	NA
[Fernandez et al., 2008]	3	S	3	C	C	C	NA	NA
[Vezzani and Cucchiara, 2010]	2	NA	1	NA	NA	S	NA	NA
Proposed	4	S	U	C	C	C	C	C

Table 2.2: Summary of main existing approaches for explicit representation of events (NA: Not Addressed; U: Unlimited; S: Simple; C: Complex).

2.3 Overview of the proposed representation model

To overcome the mentioned limitations of the models found in the reviewed literature, we propose an ontology-based hierarchical model for representing and annotating the high-level semantic information of video sequences. It has been structured in two parts: the basic ontology (composed of the basic concepts and their specializations) and the domain specific extensions. The former specifies the common fundamentals for the descriptive capabilities of the representation. The latter has to be defined for modeling each specific domain based on the expert knowledge.

We have selected the widespread adopted OWL ontology language[2] for building our proposal. This choice is motivated by the availability of a multitude of OWL editors, such as Protegé[3], Ontolingua[4] and Ontostudio[5], and the possibility of combining OWL definitions with rule languages to encode rule-based knowledge [Snidaro et al., 2007a].

[2] http://www.w3.org/2004/OWL/
[3] http://protege.stanford.edu/
[4] http://www.ksl.stanford.edu/software/ontolingua/
[5] http://www.ontoprise.de/en/products/ontostudio/

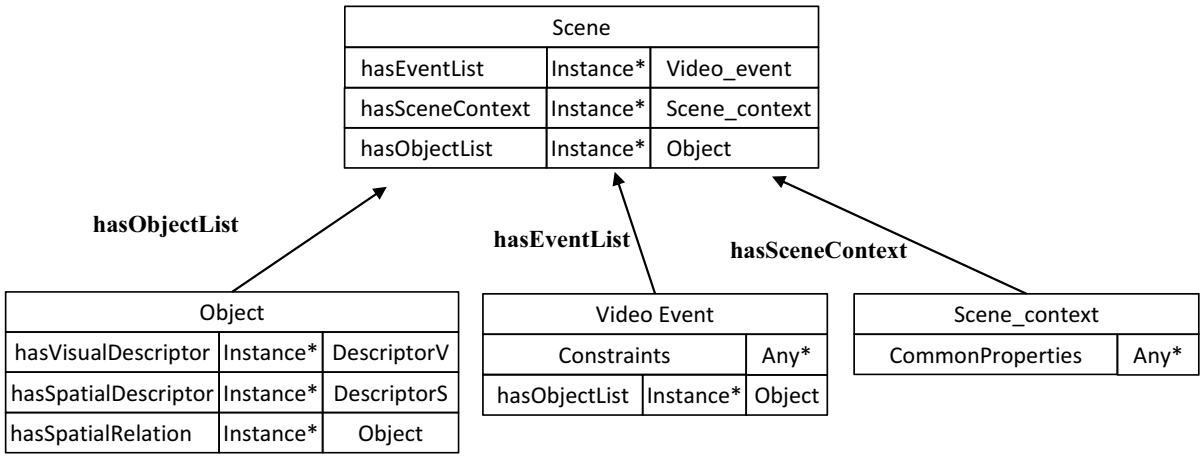


Fig. 2.1. *Scene* entity and its relations with domain knowledge entities.

The novelty of this proposal relies in the integration of three sources of high-level knowledge: *Scene*, *System* and the *User*. *Scene* regards the knowledge about a specific domain in terms of the different objects that may appear, the events that may happen and specific contextual rules. *System* consists on the self knowledge about the visual tools that analyze the video content in terms of the existing features, available algorithms, analysis schemes and reactions of the system to semantic occurrences. Additionally, the *User* source of knowledge is included to describe the consumer of the semantic descriptions (e.g., a physical person or a software program).

Thus, the basic ontology is composed by three entities that describe these sources of knowledge: the *Scene* (domain knowledge), the *System* (system or self knowledge) and the *User* (user knowledge). A more detailed description of these entities is given below:

- ***Scene***: the physical space where a real event occurs and which can be observed by one or several cameras. It includes the scene objects, their interactions (events) and the scene context. It has the following attributes (depicted in Fig. 2.1):
  - *hasObjectList*: a list of the moving and static objects that exist in the scene currently being analyzed. It describes their spatial location.
  - *hasEventList*: a list of event occurrences during the analysis of the scene. They can be of very different nature (e.g., illumination change, left baggage) and involve static and moving objects (e.g., move close to a door).
  - *hasSceneContext*: a list of contextual information descriptions that are specific for this scene. It regards specific relations between the entities *Object* and *Event*. For example, the definition of an area where only persons are expected (e.g., pedestrian area) is a contextual condition.

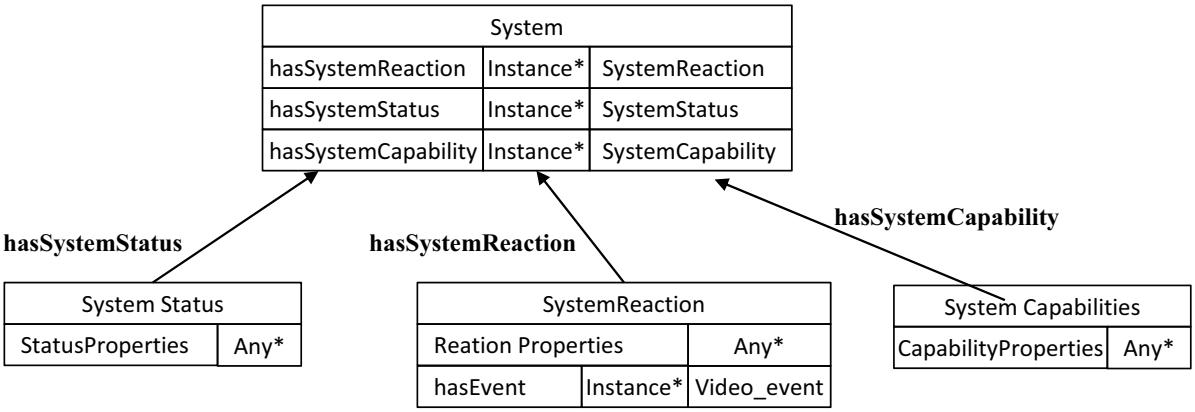


Fig. 2.2. *System* entity and its relations with self knowledge entities.

- ***System***: the visual agent that analyzes the video content. It includes the analysis capabilities of the system, the possible responses to the detected events and the system status. It has the following attributes (depicted in Fig. 2.2):
  - *hasSystemCapability*: a description of the analysis capabilities as the available algorithms, the existing features, the analysis schemes and the configurations sets.
  - *hasSystemStatus*: a description of the current system status. Depending on the application, this description can have significant variations.
  - *hasSystemReaction*: a description of the different system reactions to specific detected events or objects (e.g., start another application).

- ***User***: the final entity that manages the semantic information generated by the system. It can be a physical user, a query system or specific requirements for display purposes. It has to include a description of the interaction mode for requesting information to the system.

The proposed model provides the representation level to design cognitive systems for video analysis. Moreover, the design of such kind of systems is separated in two parts: domain-related and algorithmic-related parts. Domain experts and algorithm designers can, respectively, focus their efforts in the development of more accurate models or algorithms. In addition, this model leaves explicit the information that needs to be injected to model each application domain.

Furthermore, the three modeled knowledge sources require representations of the content that are modeling. Hence, the domain knowledge needs formal descriptions of objects, events and contextual information. The system knowledge involves modeling the parameters, algorithms, detection procedures, system reactions and status. Finally, the user knowledge demands the definition of the properties for its characterization. In the next sections, we provide all these descriptions.

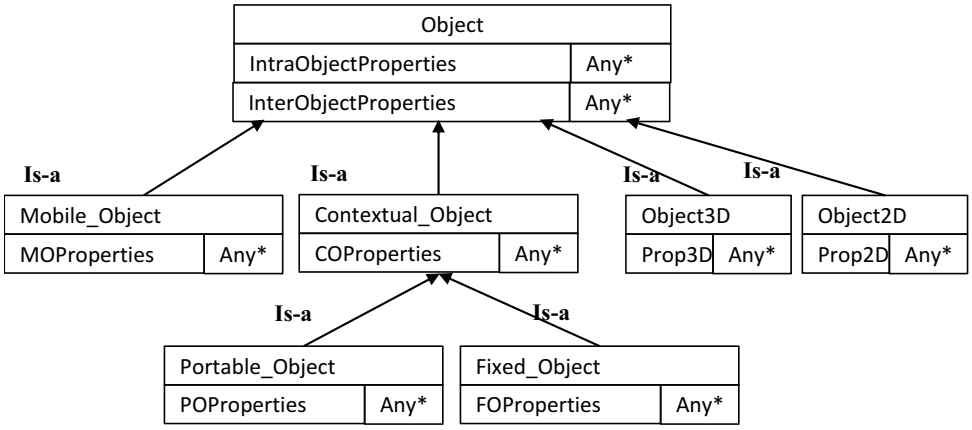


Fig. 2.3. Proposed taxonomy for the *Object* entity.

2.4 Domain knowledge description

Domain knowledge regards all the information that describes the environment in which the system operates. In our proposal, occurrences of high-level semantics are modeled through the *Scene* entity. In this section, we formalize the entities that describe this knowledge.

2.4.1 Object entity

The *Object* entity represents any real world object observed by the camera. It is the basic structural unit in the scene and most of the ontology definitions rely on the *Object* entity.

Similarly to [Francois et al., 2005], we have defined a taxonomy for the *Object* entity. Depending on the ability to initiate their own motion, we differentiate between *Mobile* and *Contextual* objects. The former regards objects that can initiate its motion (e.g., individuals, body parts, groups of people, cars). The latter relies objects that can not initiate its motion. Besides, we distinguish the *Contextual* objects between *Fixed* objects (if they cannot be displaced) and *Portable* objects (if they can be displaced). The proposed taxonomy is shown in Fig. 2.3.

Moreover, a set of properties has been defined to characterize the *Object* entity. They are divided in two types: inter-entity and intra-entity properties. We highlight the following:

- Inter-entity properties: they indicate the relationship between different entities. They are:
  - *PartOf*: indicates that the object is part of another object or has another object as part of it (e.g., fingers are part of a hand).
  - *hasSpatialRelation*: indicates the spatial relation between objects (e.g., a head is on the top of a human body).
  - *hasDetectionProcedure*: indicates the process to detect this entity (e.g., a combination of algorithms). This process has to be described as part of the system knowledge.

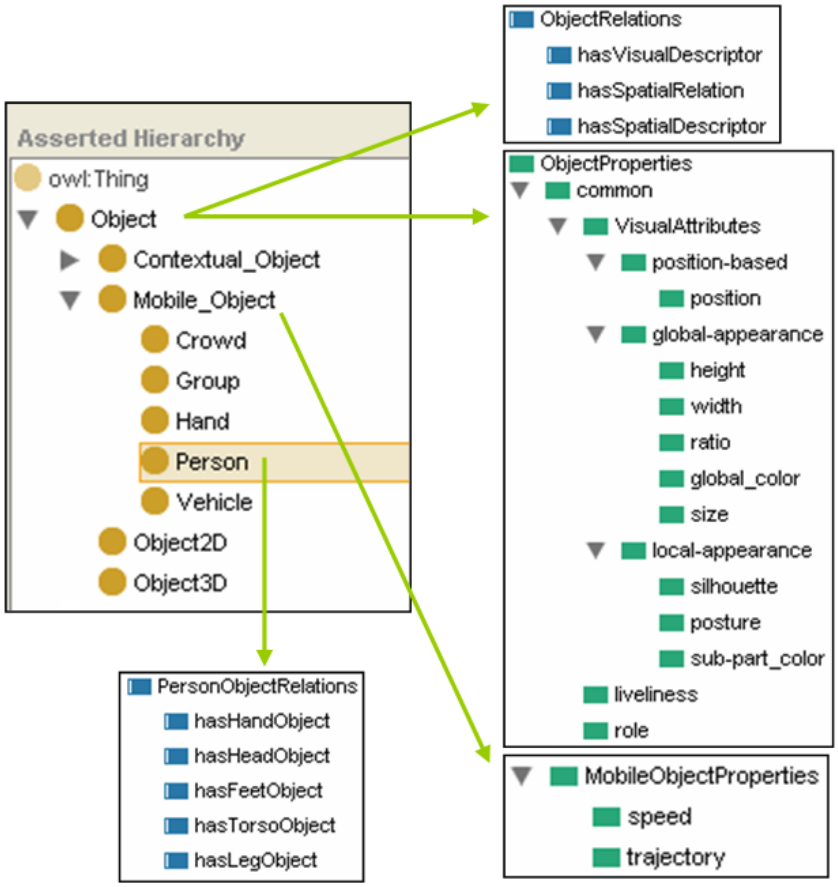


Fig. 2.4. Example of the proposed model for the *Person* entity. Its properties and relations with other entities are indicated, respectively, in green and blue color.

- Intra-entity properties: they describe the physical attributes of a specific object. They are:
  - Visual Attributes using basic types[6] such as position (*Xpos*, *Ypos*), global-appearance (*height*, *width*, and *global_ color*) and local-appearance (*shape*)
  - *hasVisualDescriptor*: indicates a visual description[7] (e.g., MPEG-7 color layout descriptor [Martinez et al., 2002]).

The different sub-entities of the *Object* entity (*Mobile* object and *Contextual* object) inherit these basic attributes adding new ones (e.g., *speed* for *Mobile* objects to characterize its mobility). In Fig. 2.4 we can see an example of a *Person* entity (sub-entity of *Mobile* object).

[6] In this document, we consider basic types in the same sense as in software programming. Thus, we refer to the *integer*, *double* and *char* types (and arrays of them) for representing entity attributes with basic types.

[7] Although we have only included visual descriptors for describing intra-entity properties, the proposed definition of the *Object* entity can be easily extended to incorporate other types of descriptors (e.g., audio descriptors)

2.4.2 Event entity

The *Event* entity describes any occurrence in the scene (e.g., activity) that is interesting for a particular purpose (e.g. content indexing, alarm generation). It ranges from low-level (e.g., illumination change) and mid-level (e.g., moving object) to high-level (e.g., leave object). Video events are defined by their objects of interest, their relations and the contextual information.

2.4.2.1 Event relations

For establishing relations between the sub-events that compose complex events, we require a formal method to express them. As proposed by [Ryoo and Aggarwal, 2009], we consider three types of event relations: temporal, spatial and logical. They are described as follows:

Temporal relations They express time dependency between the events. We consider the ones defined by Allen's temporal logic [Allen, 1994]. Each relation applies to two events and considers their corresponding time intervals. In addition, we have included the relation *same* to express concurrent events in time and the relation *duration* for event occurrences that last for a τ time period (in seconds). Let e_1 and e_2 be two events characterized by their corresponding time intervals, (t_1^{start}, t_1^{end}) and (t_2^{start}, t_2^{end}), the relations are:

$$\begin{aligned}
before(e_1, e_2) &\iff t_1^{end} < t_2^{start} \\
meets(e_1, e_2) &\iff t_1^{end} = t_2^{start} \\
overlaps(e_1, e_2) &\iff t_1^{start} < t_2^{start} < t_1^{end} \\
starts(e_1, e_2) &\iff t_1^{start} = t_2^{start} \\
during(e_1, e_2) &\iff t_1^{start} > t_2^{start} \; and \; t_1^{end} < t_2^{end} \\
ends(e_1, e_2) &\iff t_1^{end} = t_2^{end} \\
same(e_1, e_2) &\iff t_1^{start} = t_2^{start} \; and \; t_1^{end} = t_2^{end} \\
duration(e_1, \tau) &\iff t_1^{end} - t_1^{start} \geq \tau
\end{aligned} \tag{2.1}$$

Spatial relations They define geometric relations between events. Similarly to [Ryoo and Aggarwal, 2009], we use the *near* and *overlap* relations that apply to events as well as objects. Let *ent*1 and *ent*2 be two entities of the same type (e.g., events, objects) involved in the relation and ε a predefined threshold; the relations are defined as follows:

$$\begin{aligned}
near(ent1, ent2, \varepsilon) &\iff (Relative\, distance\, between\, ent1\, and\, ent2) < \varepsilon \\
overlap(ent1, ent2, \varepsilon) &\iff (Overlapping\, boundary\, between\, ent1\, and\, ent2) > \varepsilon
\end{aligned} \tag{2.2}$$

Logical relations They are devised to combine temporal and spatial relations. We consider the ones proposed by [Francois et al., 2005] (*and*, *or*, *not* and *imply*).

2.4.2.2 Event taxonomy

We combine two intrinsic characteristics of events to define the taxonomy: the number of involved entities and the temporal relation. The former allows to differentiate between single and multiple events if the event is performed by one or more *Object* entities. The latter considers simple and complex events depending if, respectively, the event can be calculated for every frame or it is composed of sub-events that span for a time period. The taxonomy contains four categories:

- *SimpleWithSingleObject* event (SSE): describes events performed by a single *MobileObject* entity and can be directly inferred from its visual attributes. These events usually correspond to changes in physical object properties (e.g., *Running*).

- *SimpleWithMultipleObject* event (SME): describes events performed by (at least two) *MobileObject* entities that can be calculated every frame (e.g., *personB-stays-inside-zoneZ)* and composed of sub-events (e.g., *Two-persons-stay-inside-a-zone).*

- *ComplexWithSingleObject* event (CSE): describes events performed by a single *MobileObject* entity and defined as a temporal combination of events. For example, the event *car-stops-near-checkpoint* is composed of the linear temporal combination of the sub-events *Car-enter*, *Car-moves-towards-checkpoint* and *Car-stops-near-checkpoint.*

- *ComplexWithMultipleObject* event (CME): describes temporal combinations of events performed by (at least two) *MobileObject* entities. The main difference with respect to the other categories is the possibility of having events with several action threads.

The proposed event categorization provides a formal method to represent events with any level of the hierarchy and define complex logical combinations of events. Fig. 2.5 presents this taxonomy.

Additionally, an event definition is composed of the following properties:

- *hasObjectList*: describes the list of *Object* entities that are involved in the event.

- *hasSub-events*: describes the sub-events that compose the event to detect.

- *hasRelations*: describes the temporal, spatial or logical relations between the objects and the sub-events that compose the event.

Fig. 2.6 illustrates some examples of event definitions for the *SimpleWithSingleObject*, *ComplexWithSingleObject* and *ComplexWithMultipleObject* event types.

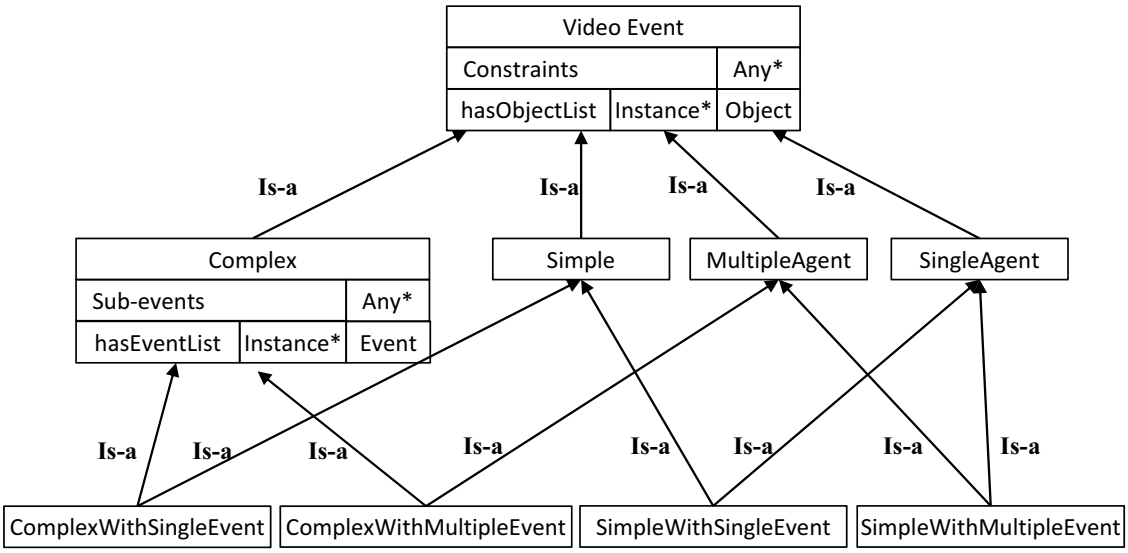


Fig. 2.5. Proposed taxonomy for the *Event* entity.

SSE_evt *Inside-zone*
object_list:
(e: entity), (z: Zone)
sub-events:
relations:
touch (e, z, 1)

SSE_evt *Far-from-object*
object_list:
(e: entity), (o: ContextualObj)
sub-events:
relations:
NOT(near (e, z, th))

(a)

CSE_evt *Abandoned-object*
object_list:
(p: Person), (o: ContextualObj)
sub-events:
e1: CSE_evt Leave_object (p, o)
e2: SSE_evt Far_from_object (p, o)
e3: CSE_evt Object_static (o)
relations:
before (e1,e2)
before (e2,e3)

(b)

CME_evt *Opposing-flow*
object_list:
(p: Person), (g: Group), (z: zone)
sub-events:
e1: SSE_evt Inside_zone (p, z)
e2: SSE_evt Inside_zone (g, z)
e3: CME_evt IsMoving (g)
e4: CME_evt IsMoving (p)
e5: SSE_evt Direction_no_similar (p,g)
relations:
same(e1,e2)
AND(e3,e4,e5)

(c)

Fig. 2.6. Event definition examples for the (a) *SimpleWithSingleObject* (SSE), (b) *ComplexWithSingleObject* (CSE) and (c) *ComplexWithMultipleObject* (CME) event types.

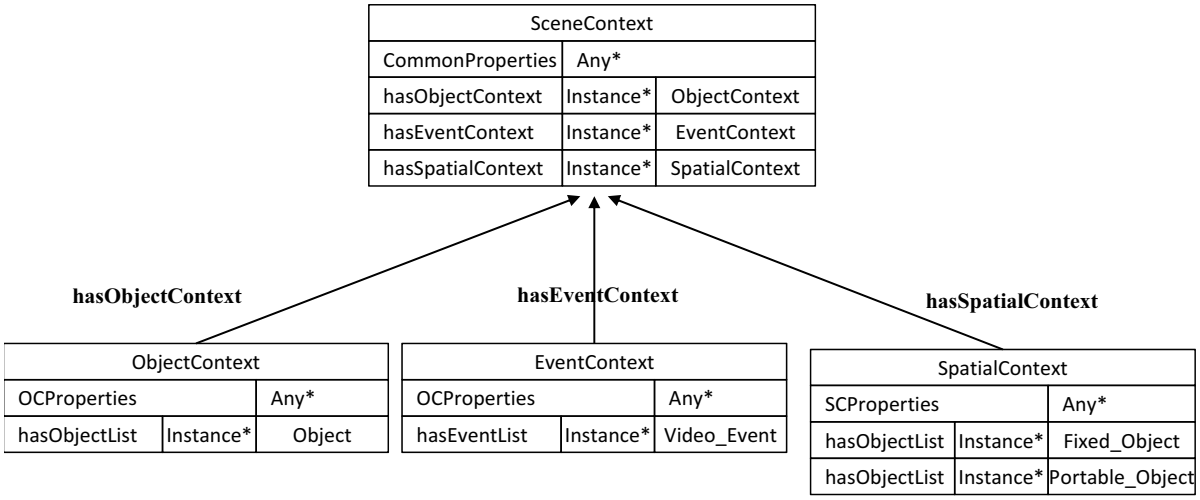


Fig. 2.7. *SceneContext* entity and its relations with domain knowledge entities.

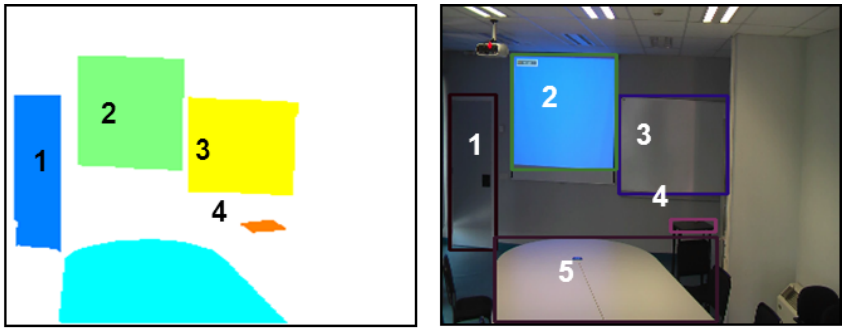
2.4.3 SceneContext entity

This entity defines all the information that may influence the way a scene is perceived and can not be described using (only) the *Object* and *Event* entities. For example, the assumption that the persons are only allowed to enter into the scene through the door is contextual information. Here, we describe two types of context for defining the properties of the scene under analysis and for modeling specific (contextual) relations between entities.

The context of a scene is defined by the following properties

- *Type*: indicates the location of the scene to be analyzed. Currently, we describe this property with two (string) values: *outdoor* and *indoor*.
- *View-distance*: indicates the distance to the observed activity of the scene. Currently, we describe this property with three (string) values: *close*, *inter* and *far*.
- *Time*: indicates the time of the scene to be analyzed. For simplicity purposes, we currently describe this property with three (string) values: *day*, *night* and *all*.
- *Crowded*: indicates if the scene to be analyzed can be considered as a crowded environment (e.g., train station). We use a Boolean value to describe this property.
- *ROI*: indicates the Region (s) Of Interest of the scene for their analysis. Currently, this information is indicated with a binary pixel mask that indicates the ROIs with the value 1.

Furthermore, the contextual relations with other types of entities are modeled through the following properties (depicted in Fig. 2.7):



(a) (b)

1. Object\ContextualObject\FixedObject\Door
2. Object\ContextualObject\FixedObject\ProjectionA
3. Object\ContextualObject\FixedObject\Blackboard
4. Object\ContextualObject\FixedObject\Table
5. Object\ContextualObject\FixedObject\Table

(c)

Fig. 2.8. Annotated scene layout example. Data correspond to (a) object tags, (b) their spatial location in the scene and (c) object categories.

- *hasSpatialContext*: defines the layout of the scene to be analyzed. This layout is composed of two layers for describing *Portable* and *Fixed* objects. Each layer describes the objects by using a textual description with an associated ID and a map indicating their spatial location. Fig. 2.8 depicts an example for the layout of *Fixed* objects.

- *hasObjectContext*: indicates specific relations between occurrences of the *Object* entity. For example, entry/exit areas of parking lot scenarios can be statically defined (using the *Object* entity) or dynamically defined by expressing a relation between them and the existing cars (as car have doors that can serve as entry/exit areas).

- *hasEventContext*: includes the event relations that can not be described using the *Event* entity. It is composed of two parts: the more likely events (in an area) and combinations among the events. For example, in the airport surveillance domain, the *unattended-luggage-detection* event could be a common event in waiting areas or typical combinations (e.g., the event *boarding-gate-door-open* is usually followed by the *people-passing-through-door*).

2.5 System knowledge description

System knowledge considers the understanding of the processes done by the module that analyzes the video content. In cognitive systems, a self system description is critical to deal with unexpected conditions (e.g., component failure) or to dynamically select the optimum algorithms for domain analysis. In this section, we formalize the entities that describe this knowledge.

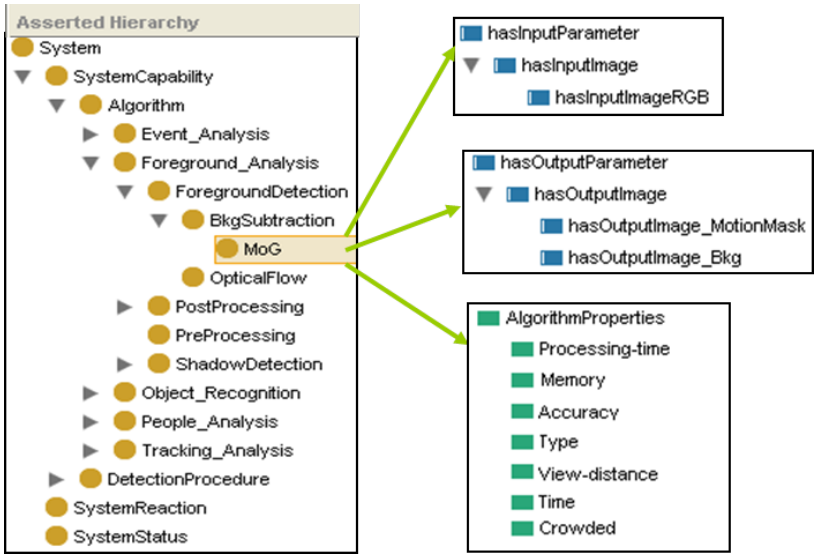


Fig. 2.9. Example of the proposed model for the *MoG* entity that describes a Background Subtraction algorithm based on Mixture of Gaussians. Its properties and relations with other entities are indicated, respectively, in green and blue color.

2.5.1 System capabilities

The *SystemCapabilities* entity defines all the available resources in the analysis system. We distinguish two different types of capabilities: the algorithms and the detection procedures. Additionally, an entity is included to define the links between algorithms (*Parameter*s).

2.5.1.1 Algorithm entity

The *Algorithm* entity represents the tools of the system that can be used for the detection of the *Object* and *Event* entities. Similarly to the *Object* entity, we have inter and intra-entity properties. Fig. 2.9 shows an example of the *Algorithm* entity. We have described the following:

- Inter-entity properties:
  - *hasInputParameterList*: indicates the input data that the algorithm needs before performing its task. This input is described by using a list of *Parameter* entities.
  - *hasOutputParameterList*: describes the output data that the algorithm generates after performing its task. This output is described by using a list of *Parameter* entities.
- Intra-entity properties:
  - Domain properties. Similarly to the *SceneContext* entity, we define some properties to describe the application domain of the algorithm. Currently, they are *Type*, *View-distance*, *Time* and *Crowded*. Their values are the same as in section 2.4.3.

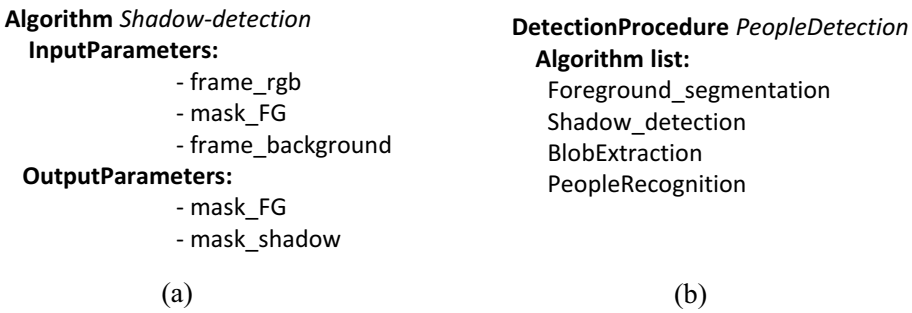


Fig. 2.10. Definition examples for the (a) *Algorithm* and (b) *DetectionProceduce* entities.

– Accuracy properties. We define some properties to characterize the accuracy of the algorithm. Currently, they are *processing-time*, *memory* and *accuracy*. Their possible values are *Low*, *Medium* and *High*.

Then, a hierarchy of the *Algorithms* is defined to represent the system capabilities. Assuming that the purpose of the system is the recognition of human-related events, we have defined the common tasks performed in this domain. They are the following: *ImageAcquisition*, *Foreground-Segmentation*, *ShadowDetection*, *Pre-Processing*, *Post-processing*, *BlobExtraction*, *PeopleRecognition*, *GroupRecognition*, *Tracking*, *FeatureExtraction* and *EventRoutines*[8]. Finally, the available algorithm implementations have to be defined as instances of these categories. The estimation of their intra-properties can be done by using training data or human expert knowledge.

2.5.1.2 DetectionProcedure entity

This entity represents the available processing schemes in the system for detecting the concepts described in the ontology. It is defined by the following inter-entity properties:

- *hasAlgorithm*: describes a list of *Algorithm* entities that are used in the detection procedure. It links at entity-level the *DetectectionProcedure* and *Algorithm* (i.e., categories) entities.
- *OrderList*: defines the usage order of the *Algorithm* entities of a *DetectionProcedure*.
- *hasTarget*: indicates the target of the processing (*Object* and *Event* entities). Thus, these entities can be linked with the *DetectionProcedure* entities by means of this property.

The input and output of each detection procedure can be inferred from the properties of its *Algorithm* entities. Fig. 2.10 depicts an example the *DetectionProcedure* and *Algorithm* entities.

[8]It should be noted that the category *EventRoutines* corresponds to simple changes of object properties (e.g., *Stopped*). More complex events, such as SSE and CSE events, are detected as a combination of them.

2.5.1.3 Parameter entity

The *Parameter* Entity represents the different inputs and outputs of the *Algorithm* entity. It is sub-categorized according to the available entities.

2.5.2 SystemReactions entity

This part defines the different system responses to the detected events or objects. We distinguish four different types: reactions to detected events (e.g., trigger an alarm, record metadata for each detected event), reactions to system failures (e.g., action to take in case of system failure), reactions to user events (e.g., user interaction with the application interface) and other type of reactions. As a first approach, we have focused on the following reactions to detected events:

- *Storage-visual-info:* indicates the recording of visual data. Three elements are defined:
  - *Path*: indicates where the metadata has to be saved by using a string type.
  - *Mode*: indicates the type of visual data to save with an integer value. Currently, the following modes are available: foreground segmentation (1), shadow removal (2), tracking (3), recognition (4) and event detection (5).
  - *Type*: indicates which type of information will be saved. Currently, two modes are possible: *continuous* (a video file) and *snapshot* (an image for each event detection).
- *Send-metadata*: indicates the ability to send metadata to a remote server. It is defined by the server information (IP address and port) and the transmission mode (e.g., TCP).
- *Storage-metadata*: indicates whether the system should save the generated metadata obtained from the analysis of the video sequence. This reaction is defined by three properties:
  - *Format*: indicates the format of the metadata to store. Currently, three types are defined: *raw* (the log of each analysis stage), *viper* (a description of the events and objects in viper format [Doermann and Mihalcik, 2000]) and *other* (for other formats).
  - *Periodicity*: indicates the frequency (in frames) for saving the metadata.
  - *Path*: indicates the location to save the metadata by using a string type.
- *Display*: represents the type of display that should be presented to the user. Currently, four display modes exist in the system, namely: *input* (for showing the input video), *object* (for showing the object information), *event* (for showing the event information) and *all* (for showing the information of each stage involved in the analysis of the video sequence).

2.5.3 SystemStatus entity

The *SystemStatus* entity generates a complete picture of the current system status. It is modeled as a set of properties such as basic settings, technical capabilities (e.g., CPU power, RAM), running processes, usage statistics (e.g., system load) and network interface status. In addition, new properties can be included depending on the system nature (e.g., distributed versus centralized).

2.6 User knowledge description

For this source of knowledge, currently, we have only described a small set of user preferences corresponding to the accuracy properties of the *Algorithm* entity. Therefore, a user may specify its preferences for *processing-time*, *memory* and *accuracy* of the system. If some properties are not specified, the highest value is assumed by default.

2.7 Summary and conclusions

This chapter has introduced the explicit representation of high-level semantics for video content. The review of the existing literature showed that there are three main approaches based on syntactic, logic and ontology-based representations. Syntactic and logic approaches offer an efficient way to encode domain knowledge but their use is limited for different, albeit related, domains and for incorporating contextual information. Generic and hierarchical representations through ontologies have been proposed as a standard, extensible and shareable model for the common knowledge about an application domain. However, existing literature lacks of a detailed description and appropriate linking at object, event and system level.

We have proposed a high-level representation model that integrates two types of knowledge related to the application domain and the analysis system. Domain knowledge involves all the high level semantic concepts for each examined domain (objects, events and context) whereas system knowledge involves the capabilities of the analysis system (algorithms, detection processes and reactions to events). This proposal defines a common model that explicitly defines the knowledge to be injected for describing each domain and system.

The proposed model helps to solve some limitations exhibited by the reviewed related work for building cognitive systems. Firstly, the detection of high-level concepts can be guided by their representation. Secondly, new inference routines can be defined to deal with unexpected conditions or data by exploiting the link between object-level events and available analysis tools. Finally, this model is suited for end-user interfaces or applications (e.g., query systems).

In the following chapters, we present how this representation model can be exploited for defining a self-configurable model for video analysis (chapter 3) and for guiding the event recognition of simple and complex events (chapter 4).

Chapter 3

A self-configurable framework for video analysis

3.1 Introduction[1]

Nowadays, advanced video analysis systems are expected to operate in a range of diverse environments within a domain allowing the on-line addition or removal, when necessary, of services and analysis capabilities [Mehmet and Choukair, 2003]. Specially, a growing demand has emerged in the video surveillance domain motivated by security issues [Remagnino and Velastin, 2006].

In this situation, the design of such systems presents many challenges related with scalability, portability and optimal resource allocation. Besides, a large number of video processing algorithms are available as an outcome of the intensive research during the past years. Thus, the selection of an algorithm to perform a particular task becomes complex as the algorithm performance depends on the operating conditions. As a result, the system may present high performance variations when applied to various environments.

Most of the current systems are hand-crafted and task-specific. They are non-scalable and their deployment in different environments requires to undergo major structural changes in many situations. Recently, some notable efforts have been made to provide modular architectures for improving scalability and portability. Nevertheless, their design is based on a human operator who has to gather a great amount of experience in many areas such as video processing, network design and data management. To reduce this requirement, some approaches propose to automate some design aspects focused on performance evaluation [Hall, 2006], available resources [Marcer-

[1] This chapter is an extended version of the publications *"J.C. SanMiguel, J. Bescós, J.M. Martínez, and A. García. DiVA: a Distributed Video Analysis framework applied to video-surveillance systems. In Proc. of IEEE Int. Workshop on Image Analysis for Multimedia Interactive Services, pp. 207-211, Klagenfurt (Austria), May 2008."* and *"J.C. SanMiguel and J.M. Martínez. Dynamic video surveillance systems guided by domain ontologies. In Proc. of IET Int. Conf. on Imaging for Crime Detection and Prevention, pp. 1-6, London (UK), Dec. 2009"*. An edited version of this chapter is under (second) review in *Machine Vision and Applications.*

ano et al., 2001] and knowledge descriptions [Greoris et al., 2007]. However, they are still not fully automatic, requiring, therefore, human intervention in most of the design stages.

In this chapter, we address the above-mentioned limitations by proposing a scalable and distributed framework for video sequence analysis that automatically estimates optimal workflows based on semantic information. Hence, this framework allows to define the analysis system for each application domain. First, a framework is designed to support the semantic-based analysis. It integrates several technologies related with data acquisition, visual analysis tools, communication protocols and data storage. Then, the application domain and their analysis capabilities are described as in chapter 2. Later, automatic workflow composition and update are proposed for analyzing each domain based on the relations between the semantic entities. Then, the optimum algorithm selection for a specific task is modeled as a constraint satisfaction problem [Apt, 2003] using the properties of the involved semantic entities. Finally, we demonstrate the success of our approach for estimating workflows for the video surveillance domain.

The remainder of this chapter is organized as follows: section 3.2 reviews the related work and section 3.3 overviews the proposed framework. In section 3.4, the workflow composition process is described and the available analysis tools are presented in section 3.5. Then, section 3.6 presents some experimental results. Finally, section 3.7 summarizes and concludes this chapter.

3.2 Related work

This chapter covers mainly the design and implementation of video analysis systems where multiple algorithms are in play; among them, video-surveillance systems are currently the most demanded for recognizing human-related events. Their design requirements are the object of very active research [Raty, 2010]. In general, the following properties are desirable: 1) scalability with load distribution, 2) real-time operation, 3) low resource consumption, 4) communication control, 5) communication over standard networks and 6) runtime re-configuration.

Traditionally, the principles for building such systems have been mainly ad-hoc and based on expert knowledge. Thus, their adaptation to different,albeit related, domains is not straightforward. Although it is generally accepted that the semantic information can improve the system performance [Maillot et al., 2004; Town, 2006], its application to video analysis is still not clear.

In the following subsections, we briefly review the existing frameworks for analysis of video events focusing on their characteristics and the control of the processing.

3.2.1 Characteristics of video event analysis frameworks

Several video analysis frameworks have been proposed by industry and academia. They are commonly described at high level not allowing a precise understanding of their capabilities. However, some generic characteristics can be studied for their classification.

One of the classical distinctions consists on their purpose: generic and specialized. For instance, [Tian et al., 2008] is focused in the video surveillance domain whereas [Venetianer et al., 2007] considers the stationary object detection in underground stations. Frameworks can be also distributed whether they divide the system tasks among autonomous processing units. Furthermore, the distribution can be static or dynamic depending on whether it allows on-line changes of the framework structure or not. For example, [Saini et al., 2009] proposed an easy insertion and removal of framework components during run-time operation. In addition, it can be distinguished whether the structural change is automatic or user-defined. For example, [Marcerano et al., 2001] automatically distributes the system tasks between the processing nodes of the network attending to the node capabilities and the task complexity. On the other hand, [Jaspers et al., 2005] requires the user interaction to add or remove new framework capabilities.

For communication issues, although most of the existing approaches use their own protocol, some approaches use standard IP-based protocols such as CORBA [Siebel and Maybank, 2004], RSTP [Carincotte et al., 2006] and SOAP [Detmold et al., 2006]. Moreover, the framework design is usually object-oriented and synchronous [Ramesh, 2005; Tian et al., 2008]. This approach can produce overhead at run-time that may cause communication bottlenecks in distributed settings. As alternative to the traditional object-oriented design, the MASCOT method [Varela and Velastin, 2004] was proposed for simplifying the communication in distributed environments.

Related literature can also be classified based on the existence of a centralized server that monitors the framework components. Initial research on this area was focused on the development of central servers to allow better management [Varela and Velastin, 2004]. However, the restriction in scalability motivated the design of decentralized frameworks by making all the framework subsystems independent and completely self-contained [Avanzi et al., 2005]. Moreover, portability and extensibility have not been explicitly considered by most of the proposals.

Finally, some efforts have been done for providing frameworks tailored to quick development of video processing applications (e.g., [Farrell et al., 2007]). However, their use and experimental validation is generally limited to simple situations.

3.2.2 Control of processing at framework level

3.2.2.1 Manual control

Some of the existing approaches propose generic and modular architectures for developing video processing applications [Avanzi et al., 2005; Tian et al., 2008; Saini et al., 2009]. They define control rules to improve scalability and portability. However, the human operator is still needed for many tasks such as the selection of the processing modules and the appropriate algorithms, as well as the specific implementation issues (e.g., resource mapping) for the different system deployments. This dependency on manual control limits their use to system developers or video

processing experts. Moreover, the complexity of manual control is increased for the analysis of dynamic environments that may require to add or remove system components during runtime.

3.2.2.2 Automatic control

Automatic control of processing aims to simplify the framework design and automate the analysis task. In the current literature, we distinguish between methods based on *Performance Evaluation* (PE), *Resource Mapping* (RM) and *Semantic Information* (SI).

PE methods compute auto-critical functions for evaluating the performance of the employed algorithms [Bins et al., 2005; Hall, 2006]. Their objective is to detect performance drops and behave accordingly (e.g., algorithm replacement, parameter adjustment). However, this evaluation is based on ground-truth data that restricts the use of the learned rules in other settings as they might not share the same data variability (and ground-truth). Furthermore, their algorithm description is focused on the input and output parameters (and their values) without containing any information about its functionality. Therefore, this control approach is semi-automatic as a human operator has to provide this information for defining the analysis workflow.

RM methods deal with the mapping of algorithms onto resources of the framework. For instance, [Marcerano et al., 2001] described the complexity and capabilities of, respectively, each task and processing node. Then, a dynamic task-node mapping is performed for the tasks requested to be completed. Similarly, [Binotto et al., 2008] proposed a reconfiguration strategy based on the load of the system processing units. The tasks are dynamically mapped onto the units that become idle. However, they do not provide solutions for adding or removing analysis capabilities. Similarly to PEs, RM methods also need the human operator to decide the structure of the task to perform and, therefore, to compose the analysis workflow.

SI approaches make use of semantics to explicitly or implicitly determine the structure of the analysis systems in the framework. *Explicit* SI approaches define semantic-based sets of rules for selecting specific algorithms in order to help the composition of workflows for video analysis. For example, [Dasiopoulou et al., 2005] described the link between objects and their recognition algorithms to make simple workflows. However, it is limited to object analysis and the execution order is manually determined. Thus, this process is semi-automatic. Similarly, [Bai et al., 2007] proposed an approach tailored to detect events for the soccer domain. Furthermore, [Greoris et al., 2007] presented a knowledge-based controlled platform for video event analysis. However, optimum algorithm selection is modeled as fine tuning (using ground-truth data) of an algorithm selected by the user. Therefore, it has the previously mentioned drawbacks. Moreover, [Nadarajan, 2010] proposed to compose workflows for simple object detection based on predefined descriptions of algorithm accuracy and user preferences. However, the structure of the workflow for each task is hard-coded and, therefore, the approach can not be automatically applied to different domains. *Implicit* SI approaches automatically learn the framework structure from

Reference	Characteristics				Control		Semantics	
	Purpose	Extensible	Portable	Distributed	Mode	Type	Storage	Use
[Marcerano et al., 2001]	Generic	NA	NA	Yes	A	R	No	No
[Siebel and Maybank, 2004]	Specific	No	No	No	M	-	No	No
[Jaspers et al., 2005]	Generic	NA	NA	Yes	M	-	Yes	NA
[Avanzi et al., 2005]	Generic	Yes	NA	Yes	M	-	Yes	NA
[Dasiopoulou et al., 2005]	Generic	NA	NA	NA	A	S	NA	Yes
[Carincotte et al., 2006]	Generic	Yes	NA	Yes	M	-	Yes	Yes
[Hall, 2006]	Specific	NA	NA	No	A	P	No	No
[Venetianer et al., 2007]	Specific	NA	NA	Yes	M	-	NA	NA
[Greoris et al., 2007]	Generic	NA	NA	NA	SA	S	Yes	Yes
[Tian et al., 2008]	Generic	Yes	NA	Yes	M	-	NA	NA
[Saini et al., 2009]	Generic	Yes	Yes	Yes	M	-	Yes	NA
[Nadarajan, 2010]	Specific	NA	NA	No	SA	S	NA	Yes
Proposed	Generic	Yes	Yes	Yes	A	S	Yes	Yes

Table 3.1: Comparative of the reviewed frameworks for video analysis. (A: Automatic; M: Manual; SA: Semi-automatic; R: Resource; S: Semantic; P: Performance; NA: Not Addressed).

semantic information. This information is usually given as a set of annotated training sequences. For example, [Town, 2006] defined the structure of a Dynamic Bayesian Network (DBN) from training sequences annotated with semantic constraints represented by an ontology.

3.2.3 Conclusion

A lack of modularization is observed in most of the proposed approaches. In addition, a specification of the tasks performed is usually not available and the framework design is usually performed by experts that develop hard-crafted solutions. Therefore, the reuse of already developed solutions is limited even for similar problems (e.g., related domains).

Automatic control of processing has been proposed to reduce the complexity of the design and deployment tasks. However, existing approaches do not provide a fully automatic solution to the problem as they do not describe in detail the relation between domain knowledge and available analysis tools (e.g., a particular combination of algorithms for detecting an event). Moreover, rule languages are applied to semantic descriptions to include the required expressiveness.

Our approach fits into the explicit SI category. Its major novelties are as follows. First, a scalable and distributed framework provides a flexible support for developing applications. Second, an automatic workflow composition is proposed to analyze different domains based on representations of domain, system and user knowledge (defined in chapter 2). Unlike existing approaches, it does not require human intervention for the automatic analysis of different domains. Table 3.1 compares our proposal against the reviewed literature.

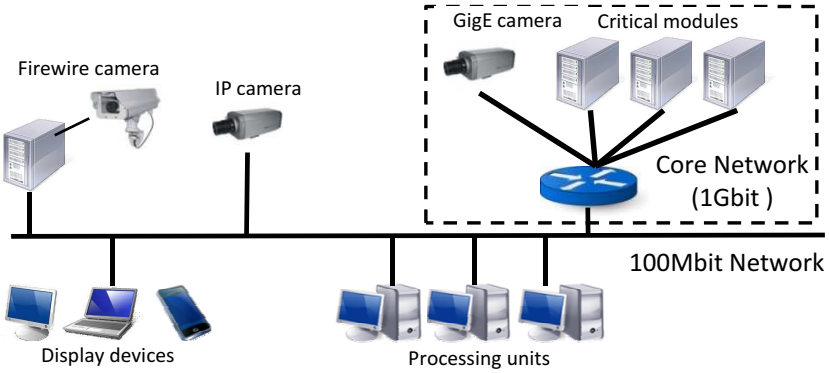


Fig. 3.1. Physical description of the proposed framework.

3.3 Framework overview

The proposed framework has the following main characteristics:

- Distributed environment for research, prototyping and deployment of visual analysis systems with support for multi-camera and semantic information.
- Modular and multi-thread design for processing at frame level.
- Asynchronous operation mode based on a client-server model.
- Dynamic workflow composition (sequential or parallel interconnection of processing algorithms) and update (on-line scalability) based on semantic information.

This framework can be abstracted in two levels (physical and logical) which are described in the following subsections.

3.3.1 Physical part

The physical part (see Fig. 3.1) is composed of the required hardware: the cameras and a cluster of standard computers (PCs) connected through a fast Ethernet network.

To cope with bandwidth restrictions and to allow operation at real-time, the framework architecture is composed of two networks. The critical framework modules are a set of rack-mounted PCs interconnected by a dedicated Gigabit Ethernet (core network). The other framework modules (mainly processing ones) are distributed in a 100BaseT Ethernet network around the core network. Different types of cameras are plugged either to an acquisition card on a PC or directly to the network for IP cameras. The processing modules are used for video acquisition, algorithm execution and data storage. The main advantage of this architecture is its flexibility. Future needs in computing power can be addressed by simply adding PCs (or replacing the oldest with more powerful ones) in the cluster.

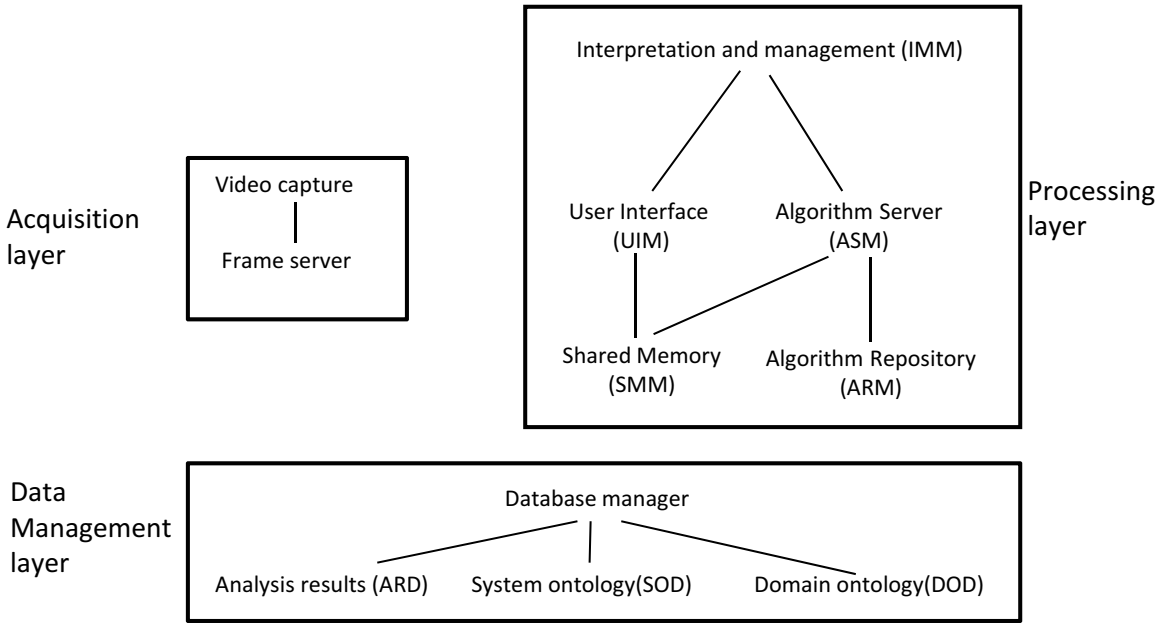


Fig. 3.2. Logical description of the proposed framework.

3.3.2 Logical part

The logical part is composed by three independent layers designed in a modular way with a specific role (see Fig. 3.2). The different modules of each layer can be distributed in several ways allowing flexible configuration. The communication is based on a client-server model; the flow control is realized through a TCP-based network. Data buffering between modules is supported at both sides for avoiding network delay problems. The system also supports the addition and removal of modules at operation time. Depending on application requirements, layers can be combined into one single component with the required functionality.

3.3.2.1 Acquisition layer

This layer acquires the video from multiple video feeds and distributes it frame-by-frame to the whole framework using a client-server model. For performance issues, the captured data is sent to a storage module in the processing layer (Shared Memory Module). Video frames are currently exchanged using baseline JPEG (ISO/IEC 10918-1) or uncompressed format. A time stamp is attached to each frame at grabbing time and is used in the processing stage (e.g., tracking algorithms). Due to its modular design, the framework can easily support the addition of new camera connection protocols by developing the corresponding video capture interfaces. Currently it handles IP, IEEE1394, GigE and USB protocols, as well as input via video files.

3.3.2.2 Data Management layer

This layer stores and distributes non-visual information required for analysis purposes or the metadata obtained by the processing layer. A database manager is included to control the use of such information. This layer is composed of three database sub-systems:

- The *Domain Ontology Database* (DOD) provides the information of the modeled application domains. Currently, it contains the domain knowledge description of chapter 2.
- The *System Ontology Database* (SOD)manages the description of the available analysis tools. Currently, it is based on the system knowledge description of chapter 2.
- The *Analysis Results Database* (ARD) stores the metadata generated by the processing modules making them available for further analysis[2]. Hence, it allows the exchange of the obtained results between processing modules in a distributed configuration.

3.3.2.3 Processing layer

In the proposed framework, a processing module is a component responsible for some particular task not related to the other layers (e.g. video analysis module, player module). The modules run concurrently and asynchronously allowing to develop distributed applications: typically each module will run on its own processor, but this is not mandatory. Moreover, some module templates have been created for easy algorithm development and integration in the framework.

This layer communicates with the Acquisition and the Data Management layers to request data (e.g., video frames, previous analysis results) and to store the obtained results. This data exchange allows the distribution of processing capabilities. Moreover, this layer includes several analysis algorithms that can be selected and combined for solving specific analysis problems.

Currently, this framework performs two tasks, ontology interpretation and video analysis, making use of the following modules (see Fig. 3.2):

- The *Interpretation and Management Module* (IMM) processes the knowledge encoded in the domain and system descriptions, then combines it with user preferences and finally requests the execution of algorithms (to the *Algorithm Server Module*).
- The *Algorithm Server Module* (ASM) provides the processing capabilities to the entire framework. It makes the visual analysis tools usable through a server.
- The *Algorithm Repository Module* (ARM) indexes the available visual analysis tools and stores their compiled versions in order to provide the processing capabilities.
- The *User Interface Module* (UIM) interacts with the content consumer (e.g., human user, software agent) to get its input (e.g., domain to analyze) and to show the obtained results.

[2]This database sub-system can be extended for developing query-based applications.

3.3.3 Analysis of a specific domain

For the analysis of video content from a specific domain, the following sequence of operations is performed:

1. *Initialization.* The UIM gets the necessary data for the analysis (e.g., domain to analyze, user preferences) and configures the IMM. Then, the IMM requests the semantic information of the domain and the analysis capabilities to the DOD and SOD modules.

2. *Semantic-based workflow composition*

   (a) The IMM requests to the ASM the analysis tools available for the selected domain by using the data indexed in the ARM. Then, the instances of the existing visual analysis tools are created and properly linked to the domain knowledge.

   (b) The IMM inspects the semantic system information to calculate the necessary resources (parameters) and to allocate memory for them in the SMM. Instances of the parameters are created and linked with the *Algorithm* instances.

   (c) The IMM interprets the system and domain semantics to select the required visual analysis tools among the available ones for domain analysis. Then, this information is sent to the ARM (via the ASM) for the creation of resources. This interpretation process is described in section 3.4.

3. *Analysis.* Finally, the IMM begins the sequential processing of the analysis workflow via execution requests to the ASM. The analysis is performed until the video file has been finished or the system is turn off (for live on-line video analysis). Results obtained by each execution are stored in the SMM that made them available for further analysis or display purposes. During run-time operation, the update of the analysis workflow (addition or removal) is performed as described in section 3.4.

3.4 Semantic-based automatic workflow composition

To overcome the current limitations exhibited by manual design based on expert knowledge, we propose to automatically compose the analysis workflow for a particular domain. This process aims to determine the optimum visual analysis tools and their associated execution order. It considers user preferences (e.g., events to recognize, system accuracy) as well as the visual tools available in the framework: algorithms (i.e., techniques for performing a task such as segmentation) and detection procedures (i.e., structured organization of algorithms for performing a task such as event recognition). For representing domain, system and user knowledge, we use the semantic descriptions defined in chapter 2. In this section, we describe the semantic relations exploited and the automatic workflow composition process.

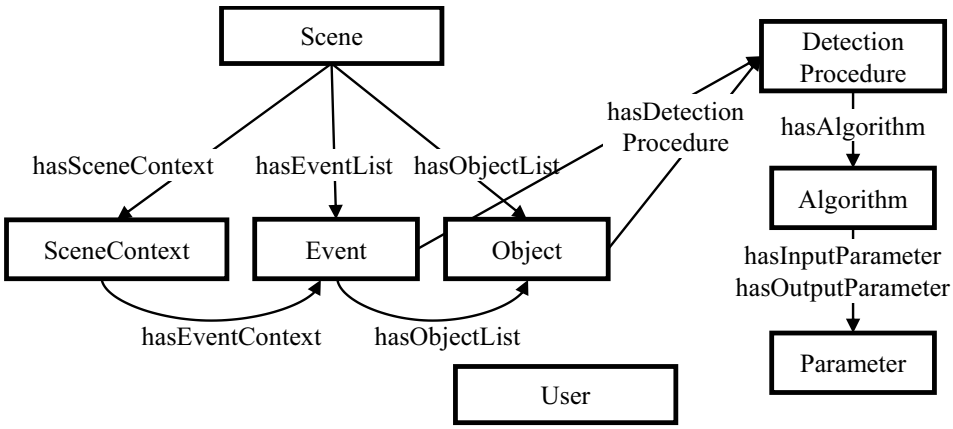


Fig. 3.3. Semantic relations exploited for automatic workflow composition.

3.4.1 Exploited entity relations

For providing such automatic composition, we study the properties of the semantic entities defined in chapter 2. In particular, we exploit the relations between the observable entities (*Object* and *Event*) and their recognition processes (*Algorithm* and *DetectionProcedure*). The key entities in this process, illustrated in Fig. 3.3, are:

- *Object* entity: represents the objects in the scene. Each *Object* entity is related to *DetectionProcedure* entities by the *hasDetectionProcedure* property.
- *Event* entity: represents any video event to be detected. Each *Event* entity is related to *DetectionProcedure* entities by the *hasDetectionProcedure* property.
- *DetectionProcedure* entity: represents the available processing schemes in the system for detecting the described semantic concepts. Each *DetectionProcedure* entity is related to *Algorithm* entities by the *hasAlgorithm* property.
- *Algorithm* entity: defines the available analysis tools. Each *Algorithm* entity is related to *Parameter* entities by the *hasInputParameter* and *hasOutputParameter* properties.
- *Parameter* entity: represents the different inputs and outputs of the algorithms available in the system. It is sub-classed according to the available algorithms.

Furthermore, two entities that describe additional information are also used:

- *UserPreferences* entity: describes the user preferences for the analysis (e.g., accuracy).
- *SceneContext* entity: defines the characteristics of the scenario (e.g., outdoor, crowded).

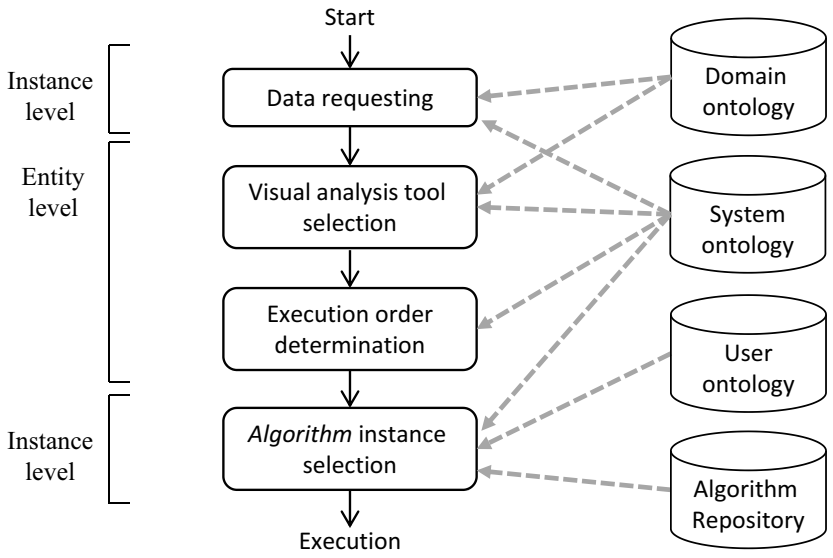


Fig. 3.4. Sequence of operations performed for semantic-based workflow composition. Gray dashed lines indicate the use of semantic information.

3.4.2 Workflow composition

Knowledge descriptions have to be properly defined prior to the composition of the workflow. Domain knowledge represents the objects and events that are expected to be observed during the analysis by means of the *Object* and *Event* entities. Instances of these entities will be created during the analysis of the domain and they are not needed for composing the workflow. System knowledge is described by the *DetectionProcedure*, *Algorithm* and *Parameter* entities. As the *DetectionProcedure* entity defines the processing schemes (and not specific implementations), there is no need to create instances. However, *Algorithm* instances are required to characterize the current system capabilities (e.g., two implementations of the foreground segmentation *Algorithm*). For the *Parameter* entity, only entity definitions are needed for the workflow composition.

The proposed approach works at entity and instance levels. It is divided in four stages: data request, visual analysis tools selection (for *DetectionProcedure* and *Algorithm* entities), execution order determination and selection of *Algorithm* instances for their execution. The sequence of operations performed is given in Fig. 3.4. They are described in the following subsections.

3.4.2.1 Data requesting

First, the framework requests data for the application domain and the user preferences. Then, instances of the *SceneContext* and the *UserPreferences* entities are created and used for composing the workflow. Furthermore, a domain description has to be available to define the entities to recognize (*Event* and *Object*). Table 3.2 shows an example of such information.

Type	Properties
SceneContext	[Type=outdoor, View-distance=far, Time=day, Crowded=No, ROI=No, hasSpatialContext=null, hasObjectContext=null, hasEventContext=null]
User	[Processing-time=high, Memory=low, Accuracy=high]
Event	[Appear, Inside-zone]
Object	[Person, Car]

Table 3.2: Example of the data requested by the framework for automatic workflow composition.

3.4.2.2 Visual analysis tools selection

This selection is directly performed by inspecting the properties of the sub-entities of the *Event* and *Object* entities defined for each domain. This stage should be considered as the integration of domain and system knowledge and it is automatically executed each time the framework is requested to analyze a specific domain. Its aim is the extraction of the needed *Algorithms,* a_i, by inspecting the *DetectionProcedures* entities associated to each *Event* and *Object* entity.

This selection process is based on rules that exploit the transitivity properties between the entities defined in the ontology. These properties define the mapping between the visual analysis tools and the relevant entities to be detected in the modeled domain. Among the available choices for rule definition, F-Logic [Dasiopoulou et al., 2005] has been selected motivated by its easy use and understanding. Firstly, three rules are defined to select all the necessary detection procedures (*DetectionProcedures* entities) to analyze a specific domain. Secondly, a fourth rule is included to extract the collection of *Algorithm* entities involved in the selected *DetectionProcedures*. This rule is applied in pairs to all the selected detection procedures. Fig. 3.5 shows these four rules. Finally, the extracted *Algorithm* entities conform the set of visual analysis tools to be executed and their execution order is computed by studying their inputs and outputs.

3.4.2.3 Execution order determination

After selecting the appropriate *Algorithm* entities for the analysis of the domain, their execution order is determined to complete the structure of the analysis workflow. This process is based on the inspection of the related *Parameter* entities (through the properties *hasInputParameter* and *hasOutputParameter* of each *Algorithm* entity). The key idea is to define different sets of input parameters, to extract the *Algorithm* entities that can be used with each set and to study their dependencies to establish an organization of the processing chain. Hence, an execution order is assigned to the group of *Algorithm* entities extracted with a specific set of parameters. For each group, the parameter relations of its *Algorithm* entities are studied to give sub-orders of

1. **IF** an *Event* e_1 has objects $O = \{o_1, ..., o_n\}$ as a part of its description **AND** objects $O = \{o_1, ..., o_n\}$ have detection procedures $DP_O = \{dp_{O1}, ..., dp_{On}\}$ respectively **THEN** E_1 *hasDetectionProcedure* $DP_{E_1} = \{dp_{O1}, ..., dp_{On}\}$.

2. **IF** an *Event* e_1 has sub-events $SE = \{se_1, ..., se_n\}$ as a part of its description **AND** sub-events $SE = \{se_1, ..., se_n\}$ have detection procedures $DP_{SE} = \{dp_{SE1}, ..., dp_{SEn}\}$ respectively **THEN** e_1 *hasDetectionProcedure* $DP_{E_1} = \{dp_{SE1}, ..., dp_{SEn}\}$.

3. **IF** an *Object* o_1 has sub-objects $SO = \{so_1, ..., so_n\}$ as a part of its description **AND** sub-objects $SO = \{so_1, ..., so_n\}$ have detection Procedures $DP_{SO} = \{dp_{SO1}, ..., dp_{SOn}\}$ respectively **THEN** o_1 *hasDetectionProcedure* $DP_{O_1} = \{dp_{SO1}, ..., dp_{SOn}\}$.

4. **IF** a *DetectionProcedure* dp_1 has algorithms $A_{DP_1} = \{a_1, a_2\, a_3\}$ and as a part of its description **AND** a *DetectionProcedure* dp_2 has algorithms $A_{DP_2} = \{a_3, a_4\, a_5\}$ as a part of its description **THEN** the set of algorithms to use is $A = \{a_1, a_2\, a_3\, a_4\, a_5\}$.

Fig. 3.5. F-logic rules for selecting the visual analysis tools through exploiting the relations between the *Object*, *Event*, *DetectectionProcedure* and *Algorithm* entities.

execution for allowing sequential and parallel processing. After that, the set of input parameters is extended with the output parameters of the examined group of *Algorithm* entities and the process is repeated with the new input set. This iterative process is performed from the minimum set of inputs, composed by the input image (named *frame-rgb* in the semantic representation model), until the list of selected *Algorithm* entities is finished. Before detailing this process, we define the following sets, algorithm types and operations on them:

Definition. $\mathbb{A}$ describes the set of *Algorithm* entities selected in the visual tool selection stage. $\mathbb{P}$ defines a generic set of *Parameter* entities. p_j and a_i describe, respectively, a *Parameter* or *Algorithm* entity. $\mathbb{A}_\mathbb{A}$, $\mathbb{A}_\mathbb{I}$ and $\mathbb{A}_\mathbb{S}$ are three sets of *Algorithm* entities for the operations of Accumulation, extraction from a set of Input parameters and the determination of the Sub-order. The execution order and sub-order are denoted by the integer variables o and s.

Definition. For *Algorithms* entities that has the same execution order, we distinguish:

- Filtering *Algorithms*: they have the same input and output
  $a_i \subset \mathbb{A}_\mathbb{S} / Input(a_i) \equiv Output(a_i)$
- Processing *Algorithms* type 1: they do not have the same input and output. Additionally, their output is contained in their input.
  $a_i \subset \mathbb{A}_\mathbb{S} / Input(a_i) \neq Output(a_i)$ AND $Output(a_i) \subset Input(a_i)$
- Processing *Algorithms* type 2: they do not have the same input and output. Additionally, their output is not contained in their input.
  $a_i \subset \mathbb{A}_\mathbb{S} / Input(a_i) \neq Output(a_i)$ AND $Output(a_i) \not\subset Input(a_i)$

Algorithm 3.1 set ofExecution order determination for the *Algorithm* entities

Input: Domain knowledge description D and selected *Algorithm* entities $\mathbb{A} = \{a_i\}$.
Output: Order o_i and sub-order s_j of each Algorithm entity a_i

begin
  Set $\mathbb{A}_\mathbb{F} = \{\emptyset\}$,$\mathbb{A}_\mathbb{S} = \{\emptyset\}$ and $o = 1$//Variable initialization
  Set $\mathbb{P} = \{frame\_rgb\}$//raw image as initial input parameter
  While $\mathbb{A}_\mathbb{S} \neq \mathbb{A}$
    $\mathbb{A}_\mathbb{I} = \{a_i \subset \mathbb{A} / Input(a_i) \equiv \mathbb{I}\}$ //select all algorithms that have determined input parameters
    if $card(\mathbb{A}_\mathbb{I}) = 1$**then**
      $AssignOrder(o, a_i)$
      $o = o + 1$
      $\mathbb{A}_\mathbb{S} = \mathbb{A}_\mathbb{S} \cup \mathbb{A}_\mathbb{I}$
    else
      Determine the type of the *Algorithm* entities (sub-order)
      $s = 1$
      for each filtering algorithm $a_j \subset \mathbb{A}_\mathbb{I}$ **do**
        $AssignOrder(o, a_j)$
        $AssignSubOrder(s, a_j)$
        $\mathbb{A}_\mathbb{S} = \mathbb{A}_\mathbb{S} \cup \{a_j\}$
      end for
      for each processing algorithm type 1 $a_j \subset \mathbb{A}_\mathbb{I}$ **do**
        $AssignOrder(o, a_j)$
        $AssignSubOrder(s + 1, a_j)$
        $\mathbb{A}_\mathbb{S} = \mathbb{A}_\mathbb{S} \cup \{a_j\}$
      end for
      for all processing algorithm type 2 $a_j \subset \mathbb{A}_\mathbb{I}$ **do**
        $AssignOrder(o, a_j)$
        $AssignSubOrder(s + 2, a_j)$
        $\mathbb{A}_\mathbb{S} = \mathbb{A}_\mathbb{S} \cup \{a_j\}$
      end for
      $o = o + 1$
    end if
    Set $\mathbb{A}_\mathbb{A} = \mathbb{A}_\mathbb{A} \cup \mathbb{A}_\mathbb{S}$ and $\mathbb{A}_\mathbb{S} = \{\emptyset\}$//Accumulate the processed algorithms in $\mathbb{A}_A$
    Set $\mathbb{P} = \{frame\_rgb, Output(\mathbb{S})\}$//Update the process input parameters
  end while
end

Definition. For operating with entities, we define the following functions:

- $Input(a_i) = \{p_j \subset \mathbb{P} / a_i \; hasInputParameter \; p_j\}$
- $Output(a_i) = \{p_j \subset \mathbb{P} / a_i \; hasOutputParameter \; p_j\}$
- $card(\mathbb{A})$ =*number of elements in the set* $\mathbb{A}$
- $AssignOrder(o, a_i) \Rightarrow$ *assigns the execution order* o *to the algorithm* i
- $AssignSubOrder(s, a_i) \Rightarrow$ *assigns the execution suborder* s *to the algorithm* i

The full execution order determination procedure is described in the Algorithm 3.1.

3.4.2.4 *Algorithm* instance selection

After composing the workflow at entity level, existing algorithms are chosen for the analysis. This selection uses knowledge of the application domain (*SceneContext* instances), the user-imposed constraints (*UserPreferences* instances) and the available capabilities (*Algorithm* instances).

We model this selection as a constraint satisfaction problem (CSP) [Apt, 2003]. We define this problem as a triple $\langle X, D, C\rangle$ where X is a super-set to define the properties of the *Algorithm* instances. It is composed of the sets $X_A = \{x_1, ..., x_N\}$ and $X_D = \{x_1, ..., x_M\}$ that, respectively, describe the domain and accuracy related properties. $D = \{d_1, ..., d_{N+M}\}$ is the set of $N + M$ domain values for each property (i.e., possible values). Hence, the properties of an *Algorithm* instance j of an entity i, a_{ij}, are defined as a mapping $V_{ij} : X \rightarrow D$. C represents the user constraints and each one is modeled as a pair $\langle T, R\rangle$ where T is a $(M + N)$ set of properties (i.e., the intra-properties of the *SceneContext* and the *User* entity) and R is a $(M + N)$-ary relation on D. We assume the definition of all the properties of the *Algorithm* instances and the constraints (i.e., the sets X and T have the same number of elements), and their same listing order. Furthermore, we consider one constraint for each application domain to be analyzed.

For solving this problem, classic CSP approaches search for a solution that satisfies all the constraints. We propose a flexible approach to measure the degree of satisfaction. The objective is to find the *Algorithm* instance that minimizes a global scoring function F defined as follows:

$$Score_{ijd} = F(V_{ij}, C), \tag{3.1}$$

where V_{ij} defines the valued properties of the instance j of the *Algorithm* entity i (a_{ij}) and C describes the constraint in terms of the domain properties and the user preferences. For a more readable notation, we have omitted the sub-indexes i and j for describing an *Algorithm* instance, and used V instead of V_{ij}. Moreover, we also use v_m instead of $v_m(V)$ for representing the m property of the *Algorithm* instance V. Then, the global function F is defined as follows:

$$F(V, C) = \sum_{m=1}^{m=M} f_d(v_m, c_m) + \sum_{n=M+1}^{n=M+N} f_a(v_n, c_n), \tag{3.2}$$

where f_d and f_a are the local scoring functions for, respectively, the domain and the accuracy properties; V and C describe, respectively, the *Algorithm* instance and the constraint; v_m and c_m are their domain properties; and v_n and c_n define, respectively, their accuracy properties.

The domain local scoring function assigns a score considering the domain properties of the *Algorithm* instance and the constraint. It is defined as follows:

$$f_d(v_m, r_m) = \begin{cases} 0 \quad if \quad v_m = c_m \quad OR \quad c_m = \{any\} \quad OR \quad v_m = \{any\} \\ 1 \quad if \quad v_m \neq c_m \end{cases} \tag{3.3}$$

For the accuracy local function, we first transform the possible property values (*Low*, *Medium* and *High* as defined in section 2.4) into a scalar domain using the following relation:

$$s(d_n) = \begin{cases} 1 & if \quad d_n = \{Low\} \\ 2 & if \quad d_n = \{Medium\} \\ 3 & if \quad d_n = \{High\} \end{cases} \tag{3.4}$$

Then, the local scoring function is defined as:

$$f_a(v_n, r_n) = \begin{cases} 1 & if \quad s(v_n) - s(r_n) > 0 \\ 0 & if \quad s(v_n) - s(r_n) \leq 0 \end{cases} \tag{3.5}$$

where $f_a(.)$ assumes that the value property *High* is the worst case and the value *Low* is the best case. This assumption is only true for the *processing-time* and *memory* properties. However, it has the opposite meaning for the *Accuracy* property (*High* value is the best case). In this situation, we just switch the conditions to invert the result of the scoring function.

Finally, instance selection is performed by using the minimum a posteriori criterion:

$$V_i^{sel} = \underset{j}{argmin}(Score_{ijd}), \tag{3.6}$$

where $Score_{ijd}$ is the satisfaction degree of the instance j of the *Algorithm* entity i for the application domain d.

Fig. 3.6 depicts an example of the instance selection process of an *Algorithm* entity. Four instances are available with different domain and accuracy properties (see Fig. 3.6(a)). Then, the information corresponding to the *SceneContext* and *UserPreferences* entities is provided for a specific domain (D0) and user (U0) as shown in Fig. 3.6(b). Finally, the method C is chosen after applying the global scoring function as it gives the minimum score (see Fig. 3.6(c)).

3.4.3 On-line workflow update

The proposed framework allows to add and remove tools (i.e., algorithms) into the analysis workflows. These operations are requested by the user or other third party applications (e.g., for failure monitoring). An insertion is performed by adding the new data (domain or system knowledge), creating the corresponding instances and computing the execution order of each added tool (if required). If new capabilities are introduced for an existing *Algorithm* entity (i.e., instances), the scoring function (Eq. 3.1) is applied to its instances for deciding the replacement of the current one being used. For new domain knowledge (e.g., new events to detect), the entire process is repeated to create a new workflow. Similarly, the removal operation differs if it affects to domain or self knowledge. A removal of domain knowledge also requires to recompose the

Instances of	Domain properties				Accuracy properties		
Algorithm i	Type	View	Time	Crowded	Time	Memory	Accuracy
Method A	Outdoor	Far	Day	No	Low	Low	Medium
Method B	Outdoor	Close	Day	No	Medium	Medium	Medium
Method C	Indoor	Inter	Day	No	Medium	High	Medium
Method D	Any	Inter	Day	Yes	High	Medium	High

(a)

Domain	Scene Context			
	Type	View	Time	Crowded
D0	Indoor	Inter	Day	No

User	User Preferences		
	Time	Memory	Accuracy
U0	Medium	Low	Medium

(b)

Instances for	Domain scores (f_d)				Accuracy scores (f_a)			Score
Algorithm i	Type	View	Time	Crowded	Time	Memory	Accuracy	$F(V,C)$
Method A	1	1	0	0	0	0	0	2
Method B	1	1	0	0	0	1	0	3
Method C	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**
Method D	0	0	0	1	1	1	1	4

(c)

Fig. 3.6. Example for selecting an instance of the Algorithm entity i by constraint satisfaction. Data corresponds to (a) description of the *Algorithm* instances, (b) the *SceneContext* and *UserPreferences* entities and (c) scores for each instance (final selection is marked in bold).

workflow. A removed *Algorithm* instance is replaced by the next available instance with the lowest score. Its main advantage is that the remaining tools (the ones that are not removed) are not eliminated from the workflow avoiding the destruction and creation of resources. In conclusion, real-time workflow update can be achieved for adding or removing analysis capabilities.

3.5 Processing library

A library of visual analysis tools is required for domain analysis. As a first approach, we have focused on the video monitoring domain. Table 3.3 summarizes the tools currently available.

For the *Algorithm* entity, the common analysis stages have been included (foreground detection, shadow removal, blob extraction, blob tracking, people recognition, group recognition, feature extraction and event analysis). For event analysis, some routines have been added to analyze object feature changes (e.g., *Inside-zone*, *Stopped*). The recognition of complex events

Algorithm instances	Domain properties				Accuracy properties		
	Type	View	Time	Crowded	Time	Memory	Accuracy
ForegroundDetection entity							
Gamma [Cavallaro et al., 2005]	Indoor	Any	Day	No	L	L	M
GMM [Stauffer and Grimson, 1999]	Any	Any	Day	No	M	M	M
Filtering+Gamma [Cavallaro et al., 2005]	Indoor	Any	Day	Yes	L	L	M
ShadowRemoval entity							
Deterministic (HSV) [Prati et al., 2003]	Any	Any	Day	Any	L	L	M
Statistical (RGB) [Prati et al., 2003]	Any	Any	Day	Any	H	L	H
BlobExtraction entity							
Connected component (CC) [Szeliski, 2011]	Any	Any	Day	Any	L	L	H
StationaryForeground entity							
No-tracking-based [Bayona et al., 2010]	Indoor	Any	Day	Yes	M	M	H
Tracking-based [Bayona et al., 2009]	Indoor	Any	Day	No	L	M	M
BlobTracking entity							
Euclidean distance [Szeliski, 2011]	Any	I-F	Any	No	L	L	L
Kalman [Caporossi et al., 2004]	Any	I-F	Day	No	L	M	L
Meanshift [Szeliski, 2011]	Any	C-I	Day	Yes	M	M	H
PeopleRecognition entity							
Edge [Garcia-Martin and Martinez, 2010]	Any	I	Day	No	M	M	H
Ellipse [Fernandez-Carbajales et al., 2008]	Any	F	Day	No	M	L	M
Aspect ratio [Fernandez-Carbajales et al., 2008]	Any	I	Day	No	L	L	L
Ghost [Fernandez-Carbajales et al., 2008]	Any	C	Day	No	H	L	M
GroupRecognition entity							
Size-based (adaptation of [Kilambi et al., 2008])	Any	I	Any	No	L	L	L
FeatureExtraction entity							
Skin detection							
HSV-based [Dadgostar and Sarrafzadeh, 2006]	Any	I-C	Day	No	L	L	L
Adap.-HSV [Dadgostar and Sarrafzadeh, 2006]	Any	I-C	Day	Yes	H	M	M
Edge energy							
Low-grad [SanMiguel and Martinez, 2008b]	Any	I-C	Day	No	L	L	L
High-grad [SanMiguel and Martinez, 2008b]	Any	I-C	Day	No	L	L	M
Background similarity							
Histogram [SanMiguel and Martinez, 2008b]	Any	I-C	Day	No	L	L	L
Contour-based [Erdem et al., 2004a]	Any	I-C	Day	No	L	L	M

Table 3.3: Library of visual analysis tools available in the proposed framework. (F: Far; I: Intermediate; C: Close; L: Low; M: Medium; H: high).

is described in chapter 4. Then, *Algorithm* instances have been created for each available technique. For example, four instances of the *PeopleRecognition* entity are defined for the algorithms based on aspect ratio, ellipse fitting, shoulder location and edges. The *Parameter* entity has been sub-classed to define the input and output of the algorithms.

For the *DetectionProcedure* entity, we have included entities to describe the processing schemes for detecting the defined *Object* and *Event* entities. Appropriate links to *Algorithm* entities are established by using the *hasAlgorithm* property. Moreover, they are also assigned to the corresponding *Object* and *Event* entities by using the *hasDetectionProcedure* property.

Domain	*SceneContext*				*Event*	*Object*
	Type	View	Time	Crowded	entities	entities
D1	Indoor	Far	Day	Yes	Inside-zone	Person
D2	Indoor	Inter	Day	No	Leave-object	Person, ContextObj
D3	Outdoor	Far	Day	No	Abandoned-object	Person, Bag
D4	Outdoor	Inter	Day	No	Stolen-object	Person, Bag

(a)

User	User Preferences		
	Proc.Time	Memory	Accuracy
U1	Low	Low	Medium

(b)

Fig. 3.7. Input data for composing the workflow. Data corresponds to the description of (a) the *SceneContext*, *Event* and *Object* entities for each domain and (b) the *UserPreferences* entity.

3.6 Experimental results

An evaluation of the automatic workflow composition process has been carried out. This section describes the experimental setup and the achieved results.

3.6.1 Setup

To show the applicability of the proposed approach, we perform an evaluation for composing the workflow in four application domains. Additionally, the preferences of one user are also considered. Table 3.7 summarizes the information used in the experiments. Fig. 3.7(a) shows the model of four domains with different recognition tasks for objects and events. Furthermore, Fig. 3.7(b) describes the user preferences.

The framework has been implemented in C++ using the OpenCV library[3] for video analysis and in Java (only the IMM module) using the OWL Protegé API[4] for ontology management. Tests were performed on two PCs (P-IV 2.8GHz and 1GB RAM) connected via a Gigabit LAN (respectively used for ontology management and video processing).

3.6.2 Results

In this section, we describe the building process for composing the workflow for the domain D1. A similar process is performed for the other modeled domains. The resulting workflows for all the domains are described in Table 3.4.

[3] http://sourceforge.net/projects/opencvlibrary/
[4] http://protege.stanford.edu/

Algorithm entity	Domain			
	D1	D2	D3	D4
ForegroundDetection	Filtering+Gamma	Gamma	GMM	GMM
ShadowRemoval	HSV	HSV	HSV	HSV
BlobExtraction	CC	CC	CC	CC
BlobTracking	Euclidean distance	Meanshift	Euclidean distance	Meanshift
PeopleRecognition	Ellipse	Aspect ratio	Ellipse	Aspect ratio
Event Detection *Routines*	Simple	Simple	Simple and complex	Simple and complex

Table 3.4: Composed workflows for all the modeled domains. Data describes the selected instances for each *Algorithm* entity.

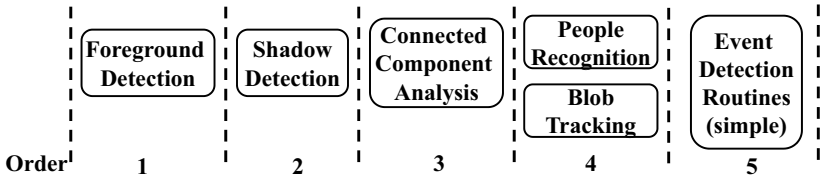


Fig. 3.8. Selected *Algorithm* entities and their execution order for the D1 domain.

3.6.2.1 Visual analysis system creation

First, the *hasDetectionProcedure* property of the *Person* and *Inside-zone* entities is examined by using the first three rules defined in subsection 3.4.2.2. Then, all *DetectionProcedures* are listed and the repeated ones are eliminated. Finally, algorithm selection is easily performed by applying the fourth rule defined in subsection 3.4.2.2 to all the *DetectionProcedures* listed. As a result of this procedure, the following *Algorithm* entities are selected: foreground detection, shadow elimination, connected component analysis, blob tracking, people recognition and the corresponding routine that models the *Inside-zone* event.

3.6.2.2 Execution order determination

As described in subsection 3.4.2.3, the *hasInputParameter* and *hasOutputParameter* properties are used to determine the execution order. First, the selected *Algorithms* that can be applied using the initial *Parameter* (i.e., *frame-rgb*) are examined. As a result, *ForegroundDetection* is selected as the first algorithm. Then, the second phase selected the *ShadowRemoval* and *BlobExtraction* algorithms. Hence, rules for collision are applied to determine that the former is applied in the first place (Rule 1) before the *BlobExtraction*. A third phase selects the *BlobTracking* and the *PeopleRecognition* algorithms. They are identified as type 2 (see subsection 3.4.2.3) so their execution order is the same (i.e., they can be executed in parallel). Finally, detection routines for the selected event are included in the last order. Fig. 3.8 depicts the final results.

Algorithm instances	Domain scores (f_d)				Accuracy scores (f_a)			Score
	Type	View	Time	Crowded	Time	Memory	Accuracy	$F(V,C)$
ForegroundDetection								
Gamma	0	0	0	1	0	0	0	1
GMM	0	0	0	1	1	0	0	2
Filtering+Gamma	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
ShadowRemoval								
Deterministic-HSV	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Statistical-RGB	0	0	0	0	1	0	0	1
BlobExtraction								
CC	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
BlobTracking								
Euclidean distance	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**2**
Kalman	0	0	0	1	0	1	1	3
Meanshift	0	1	0	0	1	1	0	3
People Recognition								
Edge	0	1	0	1	1	1	0	4
Ellipse	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**2**
Aspect ratio	0	1	0	1	0	0	1	3
Ghost	0	1	0	1	1	0	0	3
EventDetection								
Inside-zone	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Table 3.5: Algorithm instance selection for the domain D1. Data correspond to the scores for each instance (final selection is marked in bold).

3.6.2.3 Algorithm instance selection

After computing the execution order, the correct *Algorithm* instances are selected using the proposed CSP modeling as described in subsection 3.4.2.4. Table 3.5 lists the scores computed for each available instance and the selection of the ones with lower scores for execution.

3.6.2.4 Computational cost comparative evaluation

A comparison of the additional computational cost introduced by the proposed approach is listed in Table 3.6. A base workflow was been designed "by hand" with the same analysis capabilities (without any possible reconfiguration). The base framework built the workflow in approximately 4500 ms to initialize all the required resources for analysis whereas our approach took 10500 ms. An increase around 233% was observed due to the semantic-based workflow composition (approximately 6000 ms). Furthermore, this time depends on the amount of information encoded in the description of the domain and the framework capabilities. Thus, higher knowledge databases (i.e., more domain descriptions or system capabilities) will imply more delay for creating the

Approach	System creation	Frame processing	Display
Base	4500	32.4	9
Proposed	10500	33.5	9.3
Difference	+233%	+2.9%	+3.2%

Table 3.6: Computational time comparative results (ms).

workflow. However, this time could be considered as insignificant for the analysis of long sequences or 24-hour operating systems. Regarding the processing of each frame and display, no significant difference ($\sim 3\%$) was observed between both implementations.

3.7 Summary and conclusions

This chapter has described a distributed framework for video analysis that allows flexible and dynamic configuration at run-time. It provides support for acquiring, transmitting, processing and storing data (video content and metadata). Additionally, it defines a flexible environment to develop video-based applications via easy component integration.

Furthermore, we have presented how the formalization of knowledge relevant to video analysis (in terms of domain and system capabilities) is used to automatically compose and update the analysis workflow for a specific domain. This composition is based on studying the relations between the semantic entities defined for each application domain, the system capabilities and the user preferences. It is divided in four stages: data request, selection of the algorithm entities to apply, determination of their execution order and selection of the appropriate instances. A rule-based approach extracts the entities relevant for solving the analysis problem and computing their execution order. The selection of specific algorithm implementations is modeled as a constraint satisfaction problem (CSP). Experimental results demonstrated that the proposed approach was able to create different analysis workflow for each domain to be analyzed. Moreover, a comparison with a similar hand-defined workflow showed that our proposal increased the time required for system initialization whereas maintained a similar computational cost during runtime.

The main advantage of this framework is the integration of domain-based descriptions and video analysis tools. Any domain described by the semantic model of chapter 2 can be analyzed with the proposed framework. It adapts in order to analyze different domains allowing on-line reconfiguration. Besides, it can be easily extended to process multi-camera settings with multiple algorithms running in parallel, distributed over several processing modules due to its scalable nature. Moreover, the design of such kind of frameworks is separated in two parts: domain-knowledge-related and algorithmic-related parts. Domain experts and algorithm designers can focus their efforts in the development of, respectively, more accurate models or algorithms.

Chapter 4

Recognition of single-view human-related video events

4.1 Introduction[1]

Recently, the recognition of human-related events has emerged as a very promising research area due its multiple applications such as video surveillance, human-computer interaction and content-based indexing. This task presents many challenges related with the detection of people, the uncertainty of the low-level analysis (e.g., detection and tracking), the limited availability of training data, the similar appearance of different events and the modeling of complex relations.

Two main trends can be distinguished in the related literature: probabilistic and deterministic approaches. The former rely on training data to build accurate event models and they consider the uncertainty of the low level analysis. However, they are not able to model complex relations and their usage is not possible for different, albeit related, domains. The latter use rules derived from prior knowledge (e.g., semantic model). However, they do not suggest the recognition methods to use and they do not take into account the low-level uncertainty.

This chapter introduces a new approach for event recognition that takes advantage of the accuracy of probabilistic approaches as well as the descriptive capabilities of semantic-based ones. We propose a framework for complex event recognition guided by hierarchical event descriptions that can be applied to a large variety of domains. Such semantic definitions allow the decomposition of the event recognition problem into sub-tasks that can be tackled with different analysis strategies. By using the event description model defined in chapter 2, we efficiently organize the recognition methods in a two-layer structure. The first layer analyzes short-time

[1] This chapter is an extended version of the publication *"J.C. SanMiguel, M. Escudero-Viñolo, J.M. Martínez, and J. Bescós. Real-time single-view video event recognition in controlled environments. In Proc. of the Int. Workshop on Content-Based Multimedia Indexing, pp. 91-96 ,Madrid (Spain), 14-16 June 2011"*. An edited version of this chapter is under (second) review in *Computer Vision and Image Understanding*.

events by means of hierarchical Bayesian Networks (BNs) and the second layer recognizes the events that last a long period of time by using probabilistically-extended Petri Nets (PNs). The low-level uncertainty is managed by combining the probabilistic output of both layers. Moreover, problems related with the uncertainty of the semantic definitions are overcame by including the recognition problems in the event model as indicated by [Martinez-Tomas et al., 2008]. Then, the validity of the proposed approach is demonstrated by building a video analysis system to recognize human-object interactions in the video monitoring domain. Experimental results show that the system presents high accuracy for recognizing events performed by different people in diverse places whilst operating at real-time. However, a performance decrease is observed when analyzing sequences with higher complexity (e.g., crowded situations).

The remainder of this chapter is organized as follows. Section 4.2 reviews the related work. Section 4.3 overviews the event recognition framework and section 4.4 describes the two-layer recognition structure. Then, section 4.5 illustrates its application in the video monitoring domain and section 4.6 presents the experimental results. Finally, section 4.7 concludes this chapter.

4.2 Related work

Recently, several surveys have been published for human-related video event recognition [Turaga et al., 2008; Lavee et al., 2009; Poppe, 2010; Aggarwal and Ryoo, 2011; Ballan et al., 2011]. In this section, we highlight the most relevant features and methods in single-view scenarios.

4.2.1 Features

Features aim to reduce the dimensionality of the raw data contained in the video content and to represent the relevant information to the task being performed. Besides, they should be invariant over small changes of the video content in order not to limit their representation capabilities. Attending to the nature of the features, [Turaga et al., 2008] differentiates between optical flow, point trajectories, foreground blobs and filter responses. Furthermore, [Lavee et al., 2009] distinguishes between features at pixel and object level. Similarly to [Poppe, 2010] that defines global and local representations, we classify the existing features into region and point based.

Region-based features encode the observation as a whole. They rely on accurate localization of the Regions Of Interest (ROIs). Hence, they are more sensitive to viewpoint, noise and occlusions. These ROIs are usually extracted using techniques such as background subtraction [Martinez-Tomas et al., 2008], optical flow [Ali and Shah, 2010], color segmentation [Ke et al., 2010] or object detectors (e.g., people detection [Henry et al., 2003]). Then, features, such as color structure [SanMiguel and Martinez, 2008b], silhouettes [Bobick and Davis, 2001], trajectories [Medioni et al., 2001], motion patterns [Ali and Shah, 2010], 3D spatio-temporal regions [Ke et al., 2010] or human skin areas [Ayers and Shah, 2001], are derived from the ROI data.

Point-based features describe the observation as a group of independent patches extracted for each Point Of Interest (POI). This information is then combined with techniques such as bag-of-features. Opposite to region features, they are less sensitive to noise and partial occlusions and they do not need ROI extraction in most of the cases. However, the relatively small number of stable data, the high dimensionality and the high computational complexity are their main drawbacks. Among the existing approaches, we highlight the time-extensions of the Harris corner detector [Laptev, 2005], the SURF descriptor [Willems et al., 2008], the KLT point tracker [Messing et al., 2009] and the motion SIFT descriptor (MoSift) [Chen and Hauptmann, 2009].

4.2.2 Recognition methods

Many video events recognition methods have been proposed due to the maturity of the employed low-level analysis tools. Among them, a typical distinction is made between probabilistic and deterministic ones [Ryoo and Aggarwal, 2009]. Moreover, taxonomies have been proposed considering the depth [Aggarwal and Ryoo, 2011] and the duration [Turaga et al., 2008] of the temporal event structure. According to [Lavee et al., 2009], we distinguish between methods based on Pattern Recognition (PR), Graphical Modeling (GM) and Semantic Modeling (SM). Additionally, we include Hybrid Modeling (HM) to describe combination-based approaches.

4.2.2.1 Pattern Recognition methods

PR methods recognize events as a traditional classification problem in which few semantic knowledge is needed. They are simple, well understood and easy to implement. Accurate event models can be learned from training data. There are many examples such as Support Vector Machines [Schuldt et al., 2004], Nearest Neighbor [Luo et al., 2010] and Neural Networks [Huang and Wu, 2010]. Their advantages are the automatic learning of event models, the high-precision within a domain and the management of the low-level analysis uncertainty. On the other hand, they tend to increase the computational complexity, they are not able to model complex spatio-temporal relations and their usage is not possible for different, though related, domains.

4.2.2.2 Graphical Modeling methods

GM methods model the spatio-temporal event structure as a sequence of *states* by using semantic knowledge. Their descriptive capability is highly increased as compared to classic PR methods. In the current literature, we distinguish between deterministic and probabilistic approaches.

Deterministic GM These methods do not consider the uncertainty of low-level analysis and assume a fully observable state. Their structure is typically specified by expert knowledge. For instance, Finite-State-Machines (FSMs) have been proposed for modeling single activities as

a sequential ordering of states. We highlight their application for the monitoring of parking lot scenarios [Hongeng et al., 2004], aerial surveillance [Medioni et al., 2001] and object-human interactions in indoor settings [Ayers and Shah, 2001; Martinez-Tomas et al., 2008]. In addition, Petri Nets (PNs) have been proposed to coordinate multiple activities and to model relations such as sequencing, concurrency and synchronization. The PN dynamics are obtained by moving the *tokens* (i.e., video entities) between the *places* (i.e., the states) by means of the existing links with *transitions* (i.e., conditions for moving between states). A *transition* is enabled (i.e., *fired*) if all of its input *places* have at least one *token* (*hierarchical transition*) or an associated rule is satisfied (*conditional transition*).The PN approach has been successfully applied to parking lot surveillance [Castel et al., 1996] and outdoor people surveillance [Ghanem et al., 2004].

Probabilistic GM These methods take into account the uncertainty of the low-level analysis. Their structure can be efficiently learned from training data or explicitly defined. For instance, Bayesian Networks (BNs) are probabilistic methods that assume an observable state. They have been successfully applied to person interaction [Park and Aggarwal, 2004], parking lot surveillance [Hongeng et al., 2004], traffic monitoring [Kumar et al., 2005] and left luggage detection [Lv et al., 2006]. However, BNs are not able to model temporal composition of events. Moreover, Hidden Markov Models (HMMs) are proposed to combine the advantages of FSMs (temporal evolution) and BNs (probabilistic model) without assuming observable states. They have been widely used due to its simplicity and efficient parameter learning as demonstrated for event recognition based on trajectories [Cuntoor et al., 2005] and motion [Achard et al., 2008]. However, they are restricted to simple and sequential temporal patterns (Markovian model assumption) and they may fail to recognize the same event performed in a different manner (as they do not rely on semantics). Additionally, the amount of needed training data increases as the number of states and observations grow. Furthermore, Dynamic Bayesian Networks (DBNs) have been introduced to define temporal sequencing of BNs [Town, 2006]. However, DBNs present a high computational complexity and need large amounts of training data (or hand-tuning by experts).

4.2.2.3 Semantic Modeling methods

SM methods model an event as a structured semantic description specified by the domain expert. The recognition accuracy relies on the precision of its description and the extraction method. The learning of structure and parameters is not possible and they are decided by the expert. SMs are deterministic and the reasoning under uncertainty is (in general) not feasible. We distinguish two approaches: Syntactic Models (SyMs) and Constraint Satisfaction Models (CSMs).

Syntactic Models SyMs methods are widely used to represent complex events as hierarchical strings of symbols and detect them using simple routines (called primitives) such as Context-

Free-Grammar (CFG) [Bobick and Wilson, 1997]. Specifically, SyMs define the semantic events to detect (*Sentences*) using a combination of short-term events (*Words*) by means of formal rules (*Production rules*). However, SyMs present a difficulty for modeling relations more complex than just sequencing (e.g., overlap, parallelism) and for their extension and use in related domains.

Constraint Satisfaction Models CSMs methods formalize the recognition problem as a set of rules derived from the hierarchical event description such as aerial surveillance [Medioni et al., 2001], human activity analysis [Shet et al., 2005], airport monitoring [Fusier et al., 2007] and bank surveillance [Akdemir et al., 2008]. However, these methods do not consider the low-level analysis uncertainty and they are limited to the capabilities of the representation frameworks.

4.2.2.4 Hybrid Modeling methods

HM methods perform event recognition by combining the previous approaches. For example, CFG extensions for handling low-level uncertainty have been defined using HMMs [Ivanov and Bobick, 2000] and BNs [Ryoo and Aggarwal, 2009]. However, their descriptive capabilities are limited to the ones of the CFG approach and the employed analysis tools are domain specific (e.g., close-view human interactions [Ryoo and Aggarwal, 2009]). Furthermore, probabilistic extensions of PNs have been proposed for uncertain state-ordering [Albanese et al., 2008] and duration [Lavee et al., 2010b] of events. However, they do not consider uncertainty of low-level analysis. Moreover, [Lavee et al., 2010a] proposed to consider low-level uncertainty in PNs by using the particle filter framework to compute the event probability. Finally, a probabilistic extension of CSMs is proposed for complex event recognition [Romdhane et al., 2010]. However, it does not define the relation between the event descriptions and their recognition methods.

4.2.3 Conclusion

Employed features are based on using region or point data. The former are more powerful as they encode most of the information. However, they are affected by the noise and view-point. The latter are robust to these limitations but they require higher amounts of training data.

Currently, it is generally accepted that structured knowledge representations, such as deterministic GMs and SMs, can improve the event recognition performance [Town, 2006]. Unfortunately, it is not clear how to define and use these representations at the different event recognition stages; most of the existing approaches describe a small portion of this knowledge (e.g., the spatial scene layout [Ayers and Shah, 2001] and hierarchical event definitions [Fusier et al., 2007]) and model the events following a deterministic approach (i.e., defining a set of constraints) [Turaga et al., 2008]. However, they do not suggest the optimum recognition methods and they do not consider the uncertainty in both low-level analysis and semantic descriptions. Moreover, complex event representation highly increases the model size which may become difficult to manage.

Methods for event recognition	Learning phase	Observable sub-events	Uncertainty handling	Real-time analysis	Event relations		
					Nseq.	Seq.	Rec.
Pattern Recognition							
KNN [Luo et al., 2010]	A	No	Yes	Yes	No	No	No
Graphical Modeling							
BN [Park and Aggarwal, 2004]	A/M	Yes	Yes	No	No	No	No
PN [Ghanem et al., 2004]	M	Yes	No	Yes	Yes	Yes	Yes
DBN [Town, 2006]	A	No	Yes	No	No	Yes	No
FSM [Martinez-Tomas et al., 2008]	A/M	Yes	No	Yes	No	Yes	No
HMM [Achard et al., 2008]	A	No	Yes	Yes	No	Yes	No
Semantic Modeling							
SyM [Bobick and Wilson, 1997]	M	Yes	No	Yes	No	Yes	Yes
CSM [Fusier et al., 2007]	M	Yes	No	Yes	No	Yes	No
Hybrid Modeling							
Prob. PN [Albanese et al., 2008]	A/M	Yes	No	-	Yes	Yes	No
CFG-BN [Ryoo and Aggarwal, 2009]	A	Yes	Yes	No	Yes	Yes	Yes
Prob-CSM[Romdhane et al., 2010]	M	Yes	Yes	-	No	Yes	No
Proposed	A/M	Yes	Yes	Yes	Yes	Yes	Yes

Table 4.1: Comparison between the main reviewed approaches for video event recognition. (A: Automatic; M: Manual; Nseq: Non-sequential; Seq: Sequential; Rec: Recursive).

Moreover, approaches for probabilistic event recognition have shown their superior performance as compared to the deterministic ones [Lavee et al., 2009]. They learn event models from training data with high precision within a domain and allow to handle the low-level analysis uncertainty. However, most of them are computationally demanding (e.g., PRs and DBNs) and they rely on training data that may affect the accuracy of the learned event models (making them too specific or too imprecise if we use, respectively, high or low amounts of data). In addition, the representation capabilities are restricted for temporal (BNs) and complex relations (HMMs).

HMs methods combine statistical approaches (that are accurate and can handle uncertainty) and semantic-based approaches (that define structured knowledge-based models), making them a good option for robust recognition. Although this combination has not been fully explored, it is gaining attention in the recent years as a result of the above-mentioned drawbacks of the other approaches. However, current hybrid approaches partially address these problems.

Our approach fits into the HM category that takes advantage of the probabilistic accuracy as well as the descriptive capability of semantics. Unlike many existing approaches, we use a generic model for representing the event-related semantics. Then, we define a two-layer strategy for probabilistic recognition considering the low-level uncertainty as well as the semantic definitions. Table 4.1 summarizes the characteristics of the main reviewed approaches and of our proposal.

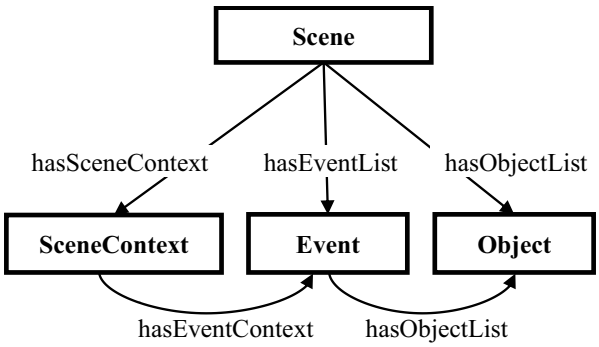


Fig. 4.1. Entity relationships exploited for event recognition.

4.3 Framework for video analysis

A video analysis framework has been designed for the event recognition task. In this section, we overview the event representation model that allows to define appropriate event recognition methods and the overall structure of the framework.

4.3.1 Event representation

Domain knowledge related to events is described by the ontology-based model defined in chapter 2. It is composed of an upper ontology that describes the structure of each knowledge type and leaves explicit the information to be inserted for modeling each domain. In particular, the *Scene* entity hierarchically represents domain knowledge in terms of the scene objects (*Object* entity), their relations (*Event* entity) and additional information (*SceneContext* entity). For achieving an effective event recognition, we propose to exploit their relations (depicted in Fig. 4.1).

The *Object* entity represents the physical scene objects. *Mobile* and *Contextual* objects are distinguished by their ability to initiate motion. Besides, *Contextual* objects are divided into *Fixed* and *Portable* objects (if they can be displaced). Therefore, events can be defined considering relations with moving entities (e.g., person), stationary objects (e.g., luggage) and scene parts (e.g., open a window). The *Event* entity represents spatio-temporal relations between *Object* entities. Each *Event* entity is related to *Object* entities by the *hasObjectList* property. Furthermore, it is sub-classed depending on the number of agents involved (single and multiple) and the temporal relation with its events (simple and complex). In this chapter, we use the latter classification to efficiently organize the event recognition methods. The *SceneContext* entity defines all the information that may influence the way a scene is perceived and can not be described using the *Object* and *Event* entities. In this chapter, we are interested in the *SpatialContext* and *EventContext* entities to provide, respectively, the scene layout and the event relations not described using the *Event* entity.

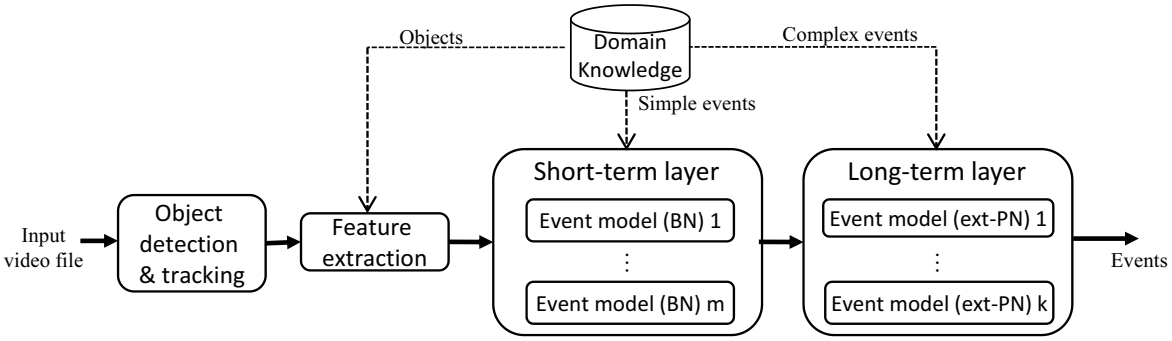


Fig. 4.2. Proposed framework for the video event recognition.

4.3.2 Framework structure

We propose an event recognition framework composed of four analysis modules (shown in Fig. 4.2). The first module detects and tracks the objects of interest (i.e., the defined *Object* entities). Then, the second module extracts the features required for event recognition. After that, a two-layer structure recognizes events considering the uncertainty of the analysis process being guided by the hierarchical event representation. The *short-term* layer detects simple events that occur during short-time periods. For each event description, a BN is defined for its detection. The *long-term* layer recognizes complex events that present a temporal relation among its sub-events. For each modeled event, a probabilistically extended PN is defined for its recognition. The proposed framework can fit the needs of a large variety of application domains by providing its knowledge description and implementing the required techniques for object detection, object tracking and feature extraction. This event recognition structure is detailed in the following section 4.4.

4.4 Hybrid recognition of events

A hybrid method is proposed to combine probabilistic analysis (for handling the low-level uncertainty) and deterministic analysis (for knowledge-based guidance of the recognition process). The objective is to establish a recognition structure for events that share similar characteristics. This section describes this structure that consists of the *short-term* and the *long-term* layers.

4.4.1 Short-term layer

The *short-term* layer recognizes the events composed of hierarchical combinations of sub-events without temporal relations (e.g., *Inside-zone*). In the representation model of chapter 2, they correspond to the simple events types *SimpleWithSingleObject* and *SimpleWithMultipleObject*. This layer formalizes the building process of the recognition method and its inference capabilities.

For modeling this layer, we have chosen the BN approach as it offers several advantages such

as the uncertainty handling and the knowledge-based structure definition. Each simple event is recognized with a hierarchical BN. The BN lower levels correspond to the observed features, the sub-events and their relations, whilst the upper level represents the event occurrence.

The BN structure is usually hard-coded relying on expert knowledge. This structure is represented by a directed acyclic graph (DAG) $\mathcal{G} = \langle \mathcal{N}, \mathcal{T} \rangle$, where $\mathcal{N}$ is the set of nodes defining the states and $\mathcal{T}$ is the set of transitions between states. We simplify the design task by proposing a formal process based on the event description model of chapter 2, where each event is represented using relations among objects (sub-events) and events (spatial and logical). Thus, some rules have to be provided for composing the BN structure from the event representation.

For each event description, a root node is included to define its recognition with a binary value (denoted with H and $\tilde{H}$ for, respectively, indicate its occurrence or not). Then, additional nodes are included into the BN based on the event description (hereafter called evidences, E_i) as follows. Firstly, sub-events or feature changes that compose the event definition are included in the network structure as nodes and are forced to produce an output probability, $P(E_i/H)$, for indicating its contribution to the modeled event. This probability can be computed using a threshold function (e.g., the probability is either 0 or 1 if a feature value is above a threshold), other learned distribution forms (e.g., Gaussian or uniform) or using the likelihood of the associated classification problem (e.g., likelihood of the people recognition task). Secondly, spatial relations among the sub-events or the objects of the definition are included in the structure as additional nodes and their probability is computed using a threshold function. Finally, logical event relations are included in the BN structure. For each logical OR relation, an additional node is included connected to the root one. Then, the transitions of the nodes that compose this relation (e.g., sub-events) are redirected to this node (they were initially connected to the root node). Thus, the compounds of this logical relation are connected to the root node through the recently included node. Its probability is computed as the maximum probability of all the incoming nodes $P(E_{OR}/H) = max(P(E_j/H))$. For each logical NOT relation, an additional node is placed between the negated sub-event and the root node. Then, its probability is computed as $P(E_{NOT}/H) = 1 - P(E_j/H)$. For the logical AND relation, no operation is performed as it is intrinsically modeled by the probability computation of the BN structure. Algorithm 4.1 summarizes this building process for the *short-term* layer and Fig. 4.3 shows the description of the event *Car-illegally-parked* (as defined in chapter 2) and the obtained BN structure.

Furthermore, the probability of a BN, composed of N variables $(H_1, ...H_N)$, is defined as a product of conditionally independent probabilities as follows:

$$P_{BN} = \prod_{i=1}^{N} P(H_i/pa(H_i)), \tag{4.1}$$

where $pa(H_i)$ is the set of parent nodes of H_i (i.e. those nodes directly connected to H_i

Algorithm 4.1 Short-term layer structure composition

Input: Domain knowledge description D.
Output: A set of BNs, $S = \{BN_i\}$, that represent the short-term layer structure based on event descriptions.

begin
$S = \{\emptyset\}$
for each simple event e_i in D **do**
  Add root node H_i to BN_i
  for each sub-event se_j of e_i **do**
    Add a node E_j in BN_i
    Connect the node E_j to H_i
  end for
  for each spatial relation sr_j of e_i **do**
    Add a node E_j in BN_i
    Connect the node E_j to H_i
  end for
  for each logical relation lr_j of e_i **do**
    Add a node E_j in BN_i
    Identify the E_k nodes that compose the relation
    Redirect the E_k node transitions to the node E_j.
    if logical relation is 'OR' **then**
      $P(E_j/H_i) = max(P(E_k/H_i))$.
    end if
    if logical relation is 'NOT' **then**
      $P(E_j/H_i) = 1 - P(E_k/H)$.
    end if
  end for
  $S = \{S,\ BN_i\}$
end for
end

via a single transition). This conditional probabilities can be learned from training data or derived from the process associated to each node. In this work, we compose a BN for each event what implies that only one variable H_i exists in each BN (hereafter called H). Therefore, the probability of each BN is limited to the computation of the $P(H/pa(H))$ term. In such case, $pa(H)$ represents the evidences $E_i, ...E_N$ of the event that are connected with the node H.

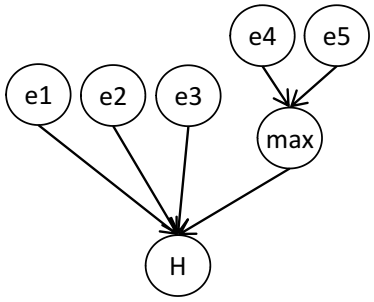
Finally, the probability of each BN, $P(H/E_{1...N})$, is computed using Bayesian inference:

$$P_{BN} = P(H/E_{1...N}) = \frac{\prod_{i=1}^{N} P(E_i/H)P(H)}{\prod_{i=1}^{N} P(E_i/H)P(H) + \prod_{i=1}^{N} P(E_i/\tilde{H})P(\tilde{H})}, \tag{4.2}$$

where H is the hypothesis (or event) and E_i are its evidences ($i = 1...N$). Similarly to [Lv et al., 2006], we assume no prior information about event occurrence ($P(H) = P(\tilde{H}) = 0.5$) and we use a default value for probabilities that are complex to estimate ($P(E_i/\tilde{H}) = 0.5$).

SSE_Event *Car-illegally-parked*
object_list:
(c: entity), (z1, z2: zone)
sub-events:
e1: SSE_evt Appears (c)
e2: SSE_evt IsCar(c)
e3: SSE_evt Stopped(c)
e4: SSE_evt InsizeZone (c, z1)
e5: SSE_evt InsizeZone (c, z2)
relations:
AND (e1, e2, e3, OR(e4, e5)

(a)



(b)

Fig. 4.3. Example for the *car-illegally-parked* event. (a) Event description in which two zones were considered for detecting a car illegally parked. (b) Proposed BN (bottom node and e_i nodes represent, respectively, the event occurrence and the described sub-events).

4.4.2 Long-term layer

The *long-term* layer recognizes events that span across frames and, therefore, they describe logical combinations of spatio-temporal relations. In the representation model of chapter 2, they correspond to the *ComplexWithSingleObject* and *ComplexWithMultipleObject* events.

For modeling this layer, we have selected the PN approach as it provides a robust formalism to express structured knowledge. According to [Ghanem et al., 2004], the PN advantages are:

- PNs can be used for both deterministic and stochastic inference of event occurrences.
- PNs provide a top-down representation for the levels of abstraction of hierarchical semantic definitions allowing sequencing, concurrency and synchronization.
- At any time, PNs can be inspected to summarize what happened in the past. This allows to incrementally recognize hierarchical events without re-evaluating past events.

Similarly to the *short-term* layer, we reduce the high dependency on expert knowledge for the design of the PN structure by using descriptions of complex events as defined in chapter 2. We recognize each complex event with a PN. Specifically, we use Plan-PNs [Albanese et al., 2008] that model the occurrence of each sub-event (opposed to Object-PNs that represent the evolution of object features in the video sequence with a unique PN [Lavee et al., 2010b]). In a Plan-PN, the *places* represent sub-events and their occurrence is indicated by a *token* in a *place*. *Transition* nodes define the conditions for the recognition of a sub-event. For modeling relations between sub-events, we employ *hierarchical* and *conditional transitions* [Ghanem et al., 2004]. Their *Firing* status allows to move *tokens* among *places*. Finally, a sink *place* is included for modeling the event occurrence. For determining the structure of each PN, we proceed as follows.

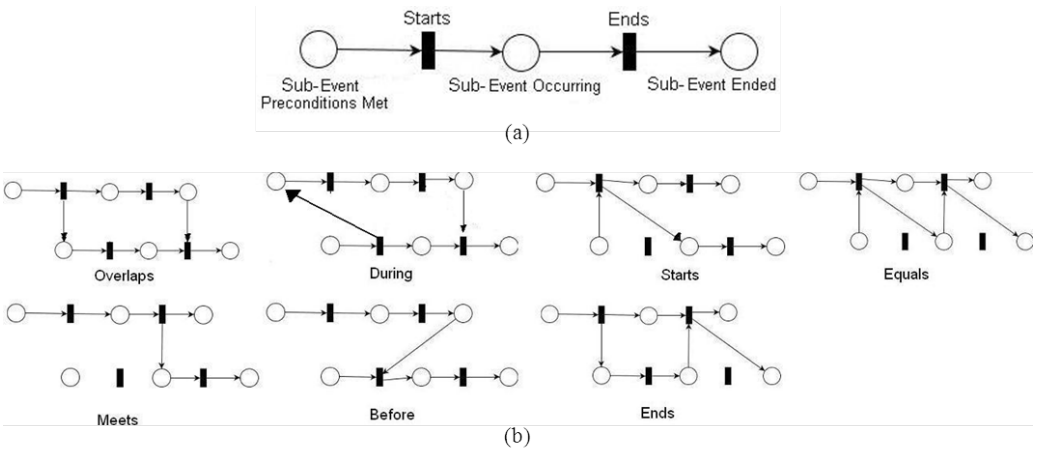


Fig. 4.4. PN models for ontology-based relations proposed by [Lavee et al., 2007]. (a) Temporal sub-event fragment. (b) Fragments corresponding to temporal relations.

Firstly, a root and a sink *places* are included in the PN structure to define the start and end of the recognition of the event. Secondly, the event description is inspected to identify its action threads defined as a sequence of executions performed by a single moving object (e.g., a car). Thirdly, each thread is processed to be included in the PN structure. For each one, two elements are included for the recognition of the moving object (e.g., car detection): a *conditional transition* and a *place*. This *transition* describes the classification problem (e.g., algorithm for car detection) and the *place* indicates its success (e.g., a car is detected). Then, the event relations of the thread (temporal, logical and spatial) are converted to the PN formalism. For the temporal relations defined in chapter 2, we use the models proposed by [Lavee et al., 2007] (depicted in Fig. 4.4), in which sub-events with temporal relation are described by using a chain of five sequential nodes (three *places* and two *transitions*) for the start and ending of the event recognition process. Then, this five-node structure is modified to fit the constraints imposed by each temporal relation. For spatial relations, a *transition* and a *place* are included in the PN to represent, respectively, this relation and its occurrence. Then, arcs are drawn from the *places* of the events that compose the relation to this additional *transition*. Logical relations are straightforward to model using the PN formalism. The logical AND relation is modeled as incoming arcs (from the events related) connected to an included *hierarchical transition*. The logical relation OR is formed as a *place* with incoming arcs from the *transitions* corresponding to the events of the relation. The logical NOT relation is defined as a condition included in the corresponding *transition* and the event probability is modified with its complementary value similarly to the *short-term* layer. Finally, the junctions between action threads are established as defined in the event description. For converting these final thread relations, we use the previously mentioned rules for event relations. Algorithm 4.2 summarizes this process.

Algorithm 4.2 Long-term layer structure composition

Input: Domain knowledge description D.
Output: A set of PNs, $L = \{PNe_i\}$, that represents the long-term layer structure based on event descriptions.

begin
$L = \{\emptyset\}$
for each complex event e_i in D **do**
  Add a root *place* PR to PNe_i
  Add a sink *place* PS to PNe_i
  for each action thread a_j of e_i **do**
    Add a *transition* T_{j1} connected to a *place* P_{j1} to PNe_i
    for moving object recognition and its success
    for each temporal relation tr_k of a_j **do**
      Add sequences of five nodes as proposed by [Lavee et al., 2007].
    end for
    for each spatial relation sr_k of a_j **do**
      Add a *transition* T_{jk} connected to a *place* P_{jk} to PNe_i
      Identify the P_n *places* of the related events
      Connect the P_n *places* with the *transition* T_{jk}
    end for
    for each logical relation lr_k of a_j **do**
      Identify the P_n *places* of the related events
      if logical relation is 'AND' **then**
        Add a *hierarchical transition* T_{jk} connected to a *place* P_{jk} to PNe_i
        Connect the P_n *places* with the *transition* T_{jk}
      end if
      if logical relation is 'OR' **then**
        Add a *place* P_{jk} to PNe_i
        Connect the P_n *places* with the *place* P_{jk}
      end if
      if logical relation is 'NOT' **then**
        Add a *transition* T_{jk} connected to a *place*P_{jk} to PNe_i
        Connect the P_n *places* with the *transition* T_{jk}
      end if
    end for
  end for
  $L = \{L, PNe_i\}$
end for
end

In addition, we overcome well-known recognition problems or uncertain definitions of events by including solutions in its definition as suggested by [Martinez-Tomas et al., 2008]. However, this operation is hard to be formalized as it relies on the expert knowledge. Subsection 4.5.3 illustrates an example of this strategy for a PN representing the *Abandoned-object* event.

Fig. 4.5 shows the description and the corresponding PN for the *Pickup-train* event. First, three *transitions* are included on the top of the PN to describe the classification stage for detecting each object involved. Then, sub-events are represented as *conditional transitions* and linked to

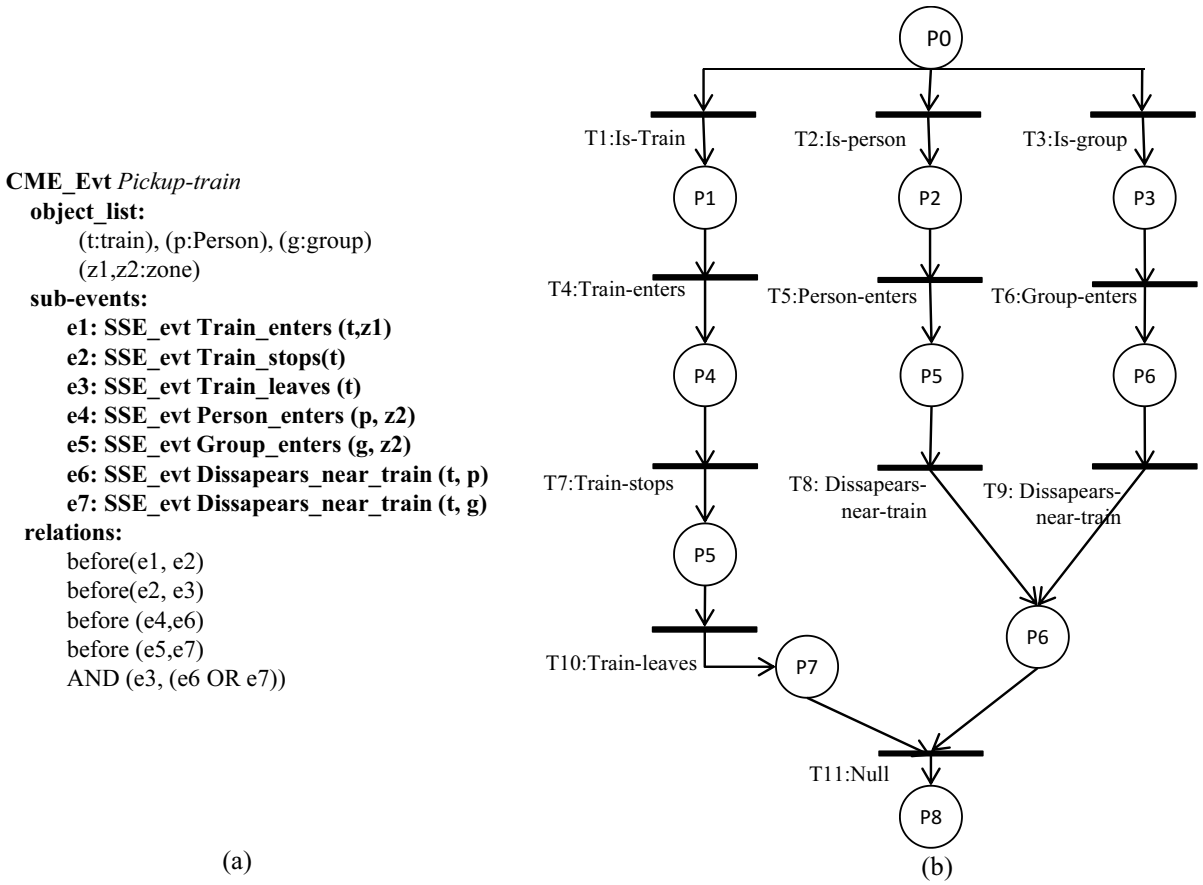


Fig. 4.5. (a) Description of the *Pickup-train* event (complex and multiple thread). (b) Proposed Petri Net (each sub-event corresponds to a conditional transition). Note that the transitions T1, T2 and T3 are included for the detection of the objects involved in the event. Besides, a hierarchical transition T11 is included at the bottom of the PN to reflect the event occurrence.

places through *arcs*. These links are guided by the relations given in the event definition. As it can be observed, the temporal relation *before* is represented using a sequential combination of *places* and *transitions*. The OR relation is represented as incoming *arcs* from two *transitions* (T8 and T9) to the P6 *place*. The event occurrence is determined by the AND combination of the two action threads. It is modeled with two incoming *arcs* from *places* P6 and P7 to *transition* T11(a *hierarchical transition* fired when all of its input *places* have at least one *token*).

Furthermore, standard PNs do not handle the analysis uncertainty as they are deterministic. As a first approach, we propose a simple combination of probabilities for considering the uncertainty of the event recognition. We assume that a probability $P(T_i)$ is obtained from each activated *transition* T_i. This probability can come from a simple event (modeled as a BN), a complex event (modeled as a PN) or the relations between them (temporal, logical and spatial). For logical relations, we compute their probability using the following rules:

$$P(T_i) = \begin{cases} \prod\limits_k P(T_k) & if \quad T_i \Longleftrightarrow AND \\ \max\limits_k (P(T_k)) & if \quad T_i \Longleftrightarrow OR \\ 1 - \frac{1}{k}\sum\limits_k P(T_k) & if \quad T_i \Longleftrightarrow NOT, \end{cases} \tag{4.3}$$

where T_i is the *transition* introduced for the logical relation, $P(T_i)$ is the resulting probability and $P(T_{1...k})$ are the probabilities of the k *transitions* connected to T_i (through *arcs* and *places*).

Traditional *transition* activation is usually performed by satisfying the associated conditions. For considering low-level uncertainty, we use a confidence level to threshold $P(T_i)$ as follows:

$$Firing_i = \begin{cases} 1 & if \quad P(T_i) > \tau_i \\ 0 & if \quad P(T_i) \leq \tau_i, \end{cases} \tag{4.4}$$

where T_i is the *transition* to be *fired* and τ_i is a threshold (confidence value). This operation reduces the computational load of the event recognition structure by discarding events with low probability that can be due to low-level analysis errors (e.g., non-accurate detections). Although a specific threshold can be defined for each *transition* based on expert knowledge (e.g., determining the error-prone analysis modules), we use the same value for all the *transitions* ($\tau_i = 0.1$).

Finally, event probability is obtained when a *token* reaches the PN sink *place* as follows:

$$P(H/T_{1...N}) = \frac{1}{N}\sum_{i=1}^{N} P(T_i), \tag{4.5}$$

where $P(H/T_{1...N})$ is the probability of the event H, $T_{1...N}$ are the *fired transitions* that the *token* has passed, N is the number of *fired transitions* and $P(T_i)$ is their probability.

4.4.3 Contextual information

Moreover, we use the *SceneContext* entity to define all the information that affects the way a scene is analyzed. It consists of the *SpatialContext, ObjectContext* and *EventContext* entities. The *SpatialContext* entity provides the initial environment layout in terms of the existing *Contextual Objects* and defines the location of events (e.g., leave objects on the *Table* object). The *ObjectContext* entity specifies relations between objects for each specific scenario (e.g., the size ratio between *Mobile* and *Contextual* objects). The *EventContext* entity describes relations between events (e.g., mutually exclusive events, predefined occurrence order for events). These context definitions are introduced to save computational cost (i.e., not analyzing events that can not happen due to the model constraints) and to decrease the false positive event rate by limiting the system response (i.e., constraining event location).

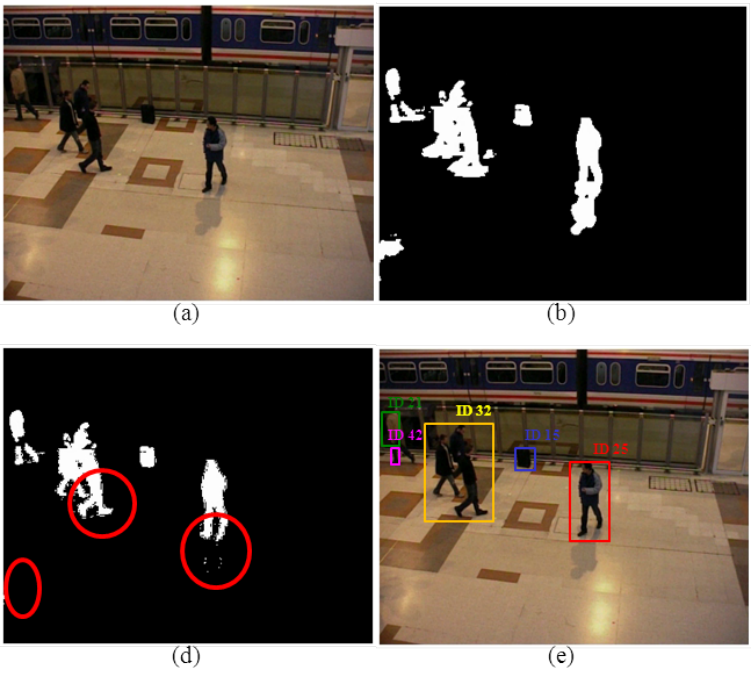


Fig. 4.6. Example the performed blob analysis. Data corresponds to (a) current image, (b) foreground mask, (c) foreground mask without shadows (marked in red) and (d) blob tracking.

4.5 Application to the video monitoring domain

We demonstrate how the proposed framework can be applied for recognizing human-object interactions in the video monitoring domain. In this section, we overview the selected tools for object detection, object tracking and feature extraction as well as the modeled events.

4.5.1 Object detection and tracking

The analysis capabilities rely on the analysis of foreground blobs[2]. First, background subtraction is applied to segment the foreground as defined in [Cavallaro et al., 2005]. Then, shadows and highlights are removed from the foreground map by analyzing the chromaticity and intensity in the HSV color space [Prati et al., 2003]. Blobs are extracted using connected component analysis. Then, blobs are tracked using a first order Kalman filter (state including position and velocity). For each frame, the filter state predicts the new positions of existing blobs and updates each one with the current blob that has the lowest Euclidean distance in terms of the centroid position and the dimensions of the blob. A unique ID identifies each tracked blob for the registration of its features through the video sequence. Fig. 4.6 shows an example of such blob analysis.

[2]In this document, we consider a blob as a connected region extracted from a binary mask that represents the foreground pixels of the scene. Observe that blob and ROI are not equivalent. We define a ROI as a region of interest in the scene that may contain several blobs.

4.5.2 Features extraction

Then, we extract the following blob-based features:

Blob velocity. This feature captures the information about blob position considering x and y axis at the same time. This is defined as follows:

$$BlobVelocity = \sqrt{v_x^2 + v_y^2}, \tag{4.6}$$

where v_x^2 and v_y^2 are, respectively, the difference between the current and previous blob mass centers in x and y axis. To get both mass centers, (x_0, y_0), we use the Hu Moments [Hu., 1962]:

$$x_0 = M_U(1,0)/M_U(0,0),\ y_0 = M_U(0,1)/M_U(0,0) \tag{4.7}$$

$$M_U(m,n) = \sum_i \sum_j x_i^m y_j^n P_{i,j},\ i,j \in Lq, \tag{4.8}$$

where $M_U(m,n)$ is the Hu Moment of order (m,n) and $P_{i,j}$ is the binary foreground value at pixel (i,j).

Blob trajectory. Trajectories of foreground blobs are used to determine their entry point, exit point and the intermediate locations in the scene.

BlobStationarity. A stationary region detector is applied to extract such regions from the video sequence. We use the hybrid algorithm proposed in [Bayona et al., 2010] that combines sub-sampling of foreground masks with motion analysis by means of the *Frame Difference* technique.

PeopleLikelihood. A people detector is applied to calculate the probability of a blob being a human. We use the edge-based model that combines four body part models (body, head, torso and legs) proposed in [Garcia-Martin and Martinez, 2010].

GroupLikelihood. We define groups considering the relative area to its closest *Contextual Object* as follows:

$$Group_b = \frac{\sum_i \sum_j P_{i,j},\ i,j \in b}{\sum_i \sum_j P_{i,j},\ i,j \in c.o}, \tag{4.9}$$

where b is the blob under analysis and co is its closer annotated object. If there is no available object annotation, the denominator is set to a value that represents the average person size in the different areas of the scene (defined as system configuration). Then, a binary probability is generated by thresholding this feature to indicate whether the blob is a group or not:

$$GroupLikelihood_b = (Group_b > th), \tag{4.10}$$

where th is a threshold. This proposal is inspired by [Kilambi et al., 2008], but avoids camera calibration by relying on the size proportion of humans and environment objects.

PeopleSkin. Assuming that hands and faces are uncovered and that the background chromaticity is different to human skin, blob skin areas are extracted using the adaptive hue thresholding proposed in [Dadgostar and Sarrafzadeh, 2006]. Skin analysis in non human blobs is avoided by just analyzing foreground blobs with high *PeopleLikelihood*.

Foregroundness. The edge energy associated with the blob boundaries is used to discriminate whether it belongs to foreground. Its computation is based on using the Canny Edge detector in the current and background images as described in [SanMiguel and Martinez, 2008b].

Backgroundness. The similarity of the blob region with its surroundings is used to discriminate if existing blobs belong to background. It is computed using the Bhattacharyya distance between the color histograms of the internal and external regions delimited by the blob contour as proposed in [SanMiguel and Martinez, 2008b].

Compactness. We compute this feature to reflect the percentage of area of active pixels (i.e., foreground) inside the blob bounding box. For each blob b under analysis, it is defined as follows:

$$Compactness_b = \frac{\sum_i \sum_j P_{i,j},\ i,j \in b}{w * h}, \tag{4.11}$$

where $P_{i,j}$ is the binary foreground value at pixel (i,j) and (w,h) are its width and height.

4.5.3 Event modeling

For this domain, we have modeled three simple and two complex human-object interactions.

The three simple events are *Leaves-object*, *Gets-object* and *Uses-object*. Their occurrence can be determined for each frame. For their definition, we have specified some simple routines that compose their representation. Thus, the *BelongToFG* routine uses the feature *Foregroundness* to indicate the degree of belonging to foreground by means of a trained Gaussian model. In a similar way, the *BelongToBG* routine uses the feature *Backgroundness* to provide a probability of belonging to the background. Moreover, the *IsOwner* routine calculates the agent (e.g., person) that is performing the event and interacting with an object (e.g., blob with low *PeopleLikelihood*). This owner is detected as the closest blob with high *PeopleLikelihood* and determined when the object appeared in the scene. The *IsPerson* routine uses the *PeopleLikelihood* feature. The *IsContextualObj* routine checks if the blob under analysis belongs to the defined *Contextual Object* entities by using the *Compactness* feature and specific appearance-based models (if the information is available). The *OverlapSkinRegion* routine calculates the spatial overlap between an entity and the skin regions of another entity. Finally, the *Stopped* routine uses the *BlobVelocity* feature to determine if a blob is moving or not. Fig. 4.7 presents their semantic descriptions and the associated (naïve) BNs. As it can be observed, additional nodes are included in the BN to represent the spatial relation *near*. Furthermore, the *Uses-object* event includes the relation *duration* that defines the length of event using the parameter *t1* to detect its occurrence.

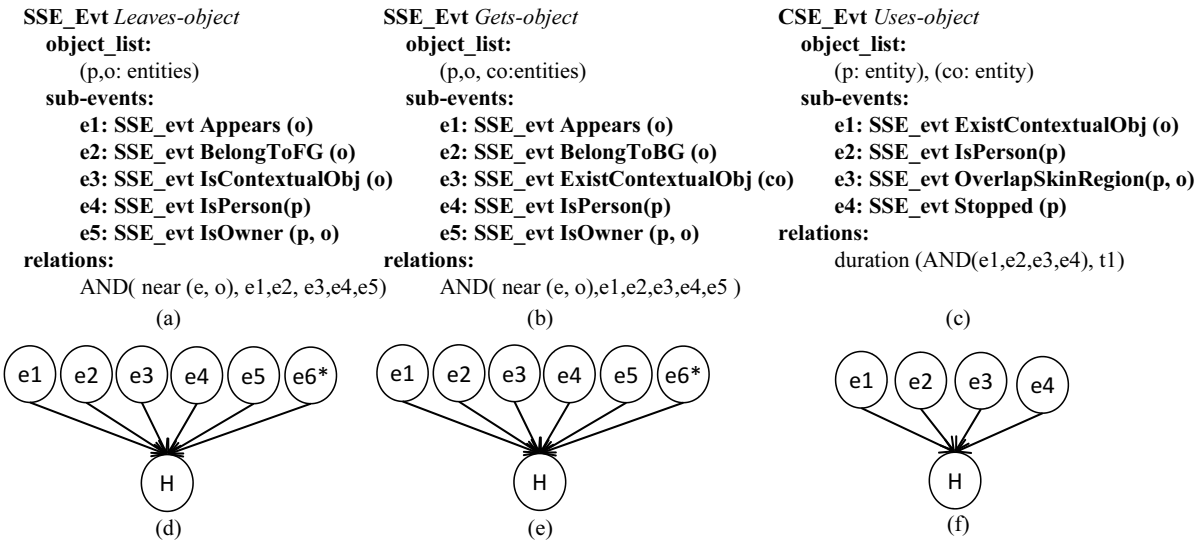


Fig. 4.7. Events models for controlled environments. Data correspond to the definition of the (a) *Leaves-object*, (b) *Gets-object* and (c) *Uses-object* events. Their naïve BN are depicted in, respectively, subfigures (d), (e) and (f). Nodes marked with * represent the spatial relation *near*.

For complex events, two common events in public video surveillance have been described: *Abandoned-object* and *Stolen-object*. Fig. 4.8 depicts the semantic model and the PN for the *Abandoned-object* event. As proposed by [Martinez-Tomas et al., 2008], we overcome known recognition problems by including strategies to solve these problems. For the *Abandoned-object* PN, the left side defines the typical model for detecting abandoned objects [Tian et al., 2010] whereas the right side describes their detection without the identification of the action owner (very common in crowded scenarios). In addition, PN loops correspond to two temporal relations: *before* (*before(e4, e5)* and *before(NOT(e5), e4)*). Besides, the *stationary* routine is defined to detect stationary objects for a given time period, described by the relation *duration(e9, 30)* (i.e., remains stationary for 30 seconds), and uses the *BlobStationarity* feature. The *Stolen-object* event definition and its associated PN are similarly defined by replacing the *Leaves-object* and *BelongToFG* events with the *Gets-object* and *BelongToBG* events.

4.5.4 Contextual information

For providing such data, we assume that two kind of environments exist in the video monitoring domain: controlled and uncontrolled. The former consists on the analysis of low-dense places where there is contextual information useful for event recognition such as the object types that can appear. The latter covers the analysis of public places that are usually crowded (e.g., train stations). They have high data variability (opposed to the controlled situation) and few contextual data is available. In this subsection, we describe the context defined for each situation.

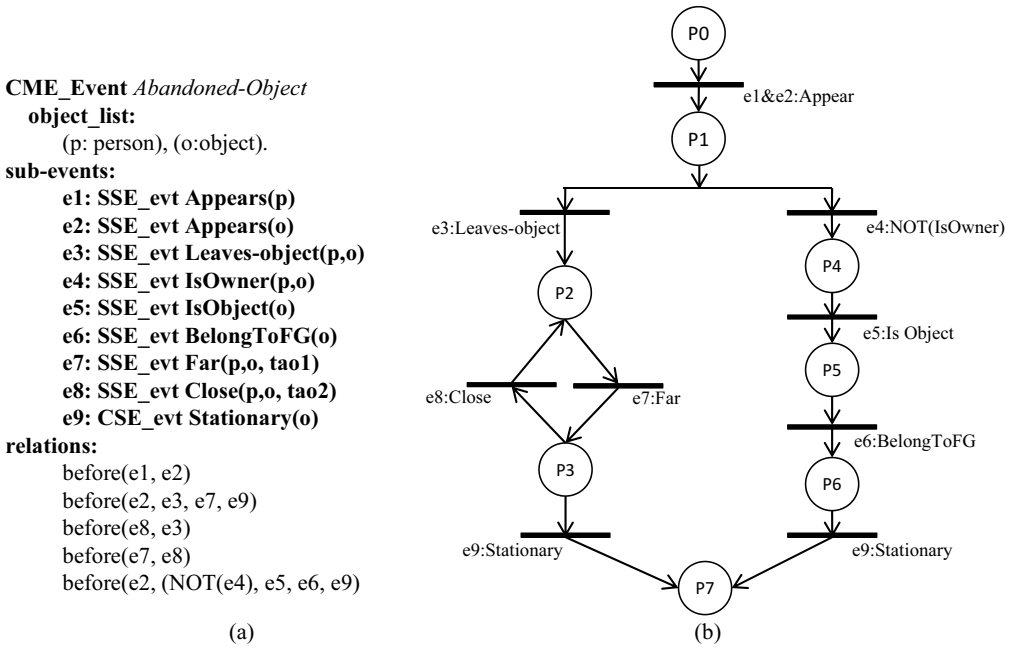


Fig. 4.8. Complex event *Abandoned-object* modeled for the uncontrolled environment. Data correspond to (a) semantic definition and (b) the corresponding PN.

4.5.4.1 Controlled environments

For the *Object* entity, we distinguish *Mobile* (*Group* and *Person*) and *Contextual Objects*. Among the latter, we discriminate between *Fixed* (*Wall, Window, Floor, Table*, *Door, Blackboard, Screen* and *ProjectionArea*) and *Portable Objects* (*Chair*, *Laptop*, *MobilePhone* and *Generic*).

Furthermore, the *SceneContext* entity is used to describe the contextual information of each scenario in this environment. *SpatialContext* provides the location of the existing *Contextual Objects*. The *ObjectContext* entity determines relations between objects for each scenario (e.g., the size ratio between *Mobile* and *Contextual* objects for detecting *Groups*). The *EventContext* entity defines relations between events (e.g., mutually exclusive events or predefined event occurrence order). Currently, we have constrained the location of the *Leaves-object* event to the *Table* areas by using the spatial relation *overlap*. Besides, the *ExistContextualObj* routine, that checks if a *Contextual* object exists in the same spatial location as the blob under analysis, is included as context for the *Gets-object* and *Uses-object* events. This constraint is included in their BNs as an additional node. It requires the knowledge of the scenario layout provided by the *SceneContext* entity (that is impossible for uncontrolled environments). These spatial constraints are introduced to decrease the false positive event rate by constraining the event location and to save computational cost by avoiding possible occurrences of complex events that include the context-modified simple events (e.g., *Abandoned-object* event includes the *Leaves-object* event).

4.5.4.2 Uncontrolled environments

Unlike in the controlled situation, little prior information is available. Although it is assumed that some types of objects are known (e.g., person, trains), the variability of the features of the objects of interest is not known (e.g., luggage appearance). Moreover, the relation of events with the layout of the scene is more difficult to estimate as, in these settings, the recording device (e.g. camera) is typically placed at a medium or long distance from the action. Therefore, the type of events that can be recognized with enough accuracy is limited, in most of the cases, to the ones related to trajectory analysis. For these reasons, objects and events have to be defined in general terms without considering specific appearances. Due to the complexity of extracting contextual information in this situation, the *SceneContext* entity is not exploited.

For the *Object* entity, we distinguish *Mobile* (*Group* and *Person*) and *Contextual*. The latter is discriminated into *Fixed* (*Wall, Window, Floor,* and *Area*) and *Portable Objects* (*Generic*).

4.6 Experimental results

We evaluate our approach on the video monitoring domain for the controlled and uncontrolled environments modeled in the previous section. It has been implemented using the OpenCV library[3]. Tests were performed on a standard PC (P-IV 2.8GHz and 2GB RAM).

For both environments, we heuristically defined the system parameters required for the object detection and tracking stages. In particular, we used the parameter $\sigma_n = 12$ for controlling the foreground detection [Cavallaro et al., 2005] and the set $\{\alpha = 0.5, \beta = 0.9, \tau_S = \tau_H = 0.2\}$ for removing shadows and highlights [Prati et al., 2003]. For feature extraction, we used the default parameter values proposed by the authors of each employed technique (see subsection 4.5.2).

4.6.1 Performance evaluation criteria

For matching event annotations and detections, we have defined the following criteria:

$$Match(E^{GT},\, E^{D}) = \begin{cases} 1 \quad if & score & > & \rho & \bigwedge \\ & \left|T^{D}_{start} - T^{GT}_{start}\right| & < & \tau_1 & \bigwedge \\ & \left|T^{D}_{end} - T^{GT}_{end}\right| & < & \tau_2 & \bigwedge \\ & \frac{2\left|A^{GT} \cap A^{D}\right|}{|A^{GT}|+|A^{D}|} & > & \sigma & \\ 0 \quad & Otherwise, & & & \end{cases} \tag{4.12}$$

where E^{GT} and E^{D} are the annotated and detected events; *score* is the probability of the detected event; $(T^{D}_{start}, T^{D}_{end})$ and $(T^{GT}_{start}, T^{GT}_{end})$ are the frame intervals of the annotated (GT) and detected (D) events; $\left|A^{GT}\right|$ and $\left|A^{D}\right|$ represent the average area (in pixels) of each event;

[3] http://sourceforge.net/projects/opencvlibrary/

Cat.	Frames	Event Occurrences			Complexity			
		LEA	GET	USE	FG	BT	FE	ED
C1	41753	28	21	9	M	L	M	M
C2	36446	25	15	10	M	M	M	H
C3	35570	29	28	35	V	H	V	V
Total	113769	82	64	54	-	-	-	-

Table 4.2: Dataset description for controlled environments. (LEA: *Leaves-object*; GET: *Gets-object*; USE: *Uses-object*; FG: Foreground seg.; BT: Blob tracking; FE: Feature Extraction; ED: Event recog.; L: Low; M: Medium; H: High; V: Very High).

$\left|A^{GT} \cap A^{D}\right|$ is their average spatial overlap (in pixels); ρ, τ_1, τ_2 and σ are positive thresholds (heuristically set to the values $\rho = 0.75$, $\tau_1 = \tau_2 = 100$ and $\sigma = 0.5$).

Then, we evaluate the recognition accuracy with the Precision (P) and Recall (R) measures. Precision is the ratio between the correct and the total number of detections. Recall is the ratio between the correct detections and the total number of annotations. They are defined as follows:

$$P = \frac{TP}{TP + FP} \quad R = \frac{TP}{TP + FN}, \tag{4.13}$$

where TP (True Positive) are the correct event detections, FN (False Negatives) are the missed events and FP (False Positive) are the wrong event detections. For event annotation and performance evaluation, the ViPER toolkit [Doermann and Mihalcik, 2000] has been used.

4.6.2 Controlled environments

For this situation, we evaluate the proposed approach for recognizing the previously defined three simple events: *Leaves-object* (LEA), *Gets-object* (GET) and *Uses-object* (USE).

4.6.2.1 Dataset

A dataset has been collected from selected sequences of the VISOR[4], the HERMES[5], the WCAM[6], the CANDELA[7] and the ED[8] public datasets. For discussing the achieved results, we have classified the sequences into three categories attending to an initial complexity estimation of the analysis stages that compose the proposed framework. Table 4.2 lists the characteristics of this dataset and Fig. 4.9 shows sample frames of selected sequences.

[4] http://www.openvisor.org/
[5] http://iselab.cvc.uab.es/indoor-cams
[6] http://wcam.epfl.ch/
[7] http://www.multitel.be/~va/candela/abandon.html
[8] http://www-vpu.eps.uam.es/EDds/

Fig. 4.9. Sample frames of the dataset for controlled environments. Rows 1, 2 and 3 correspond to categories C1, C2 and C3. From top-left to bottom-right: *AbandonedObject* (VISOR), *Indoor_activity1* (WCAM), *S1_0001* (MR), *Indoor_activity2* (WCAM), Cam1_indoor (HERMES), *S2_0004* (MR), *S3_0001* (MR), *Indoor_1.07* (CANDELA) and S3_0002 (MR).

4.6.2.2 Results

In total, our approach detected 657 event occurrences. Their probability distribution is displayed in Fig. 4.10. As it can be observed, a high number of events are detected with low probability ($score < 0.1$). They can be easily discarded as most of them correspond to false detections. However, the events with intermediate probability ($0.2 < score < 0.8$) present high uncertainty as it is difficult to decide whether they are correct or not. Low event probability can be due to non-accurate low-level analysis or event occurrences that can not be described with their semantic models (i.e., event model inconsistencies). Finally, we filtered the initial detections by thresholding the event probability with $\rho = 0.75$ (see Eq. 4.12), obtaining 183 event occurrences.

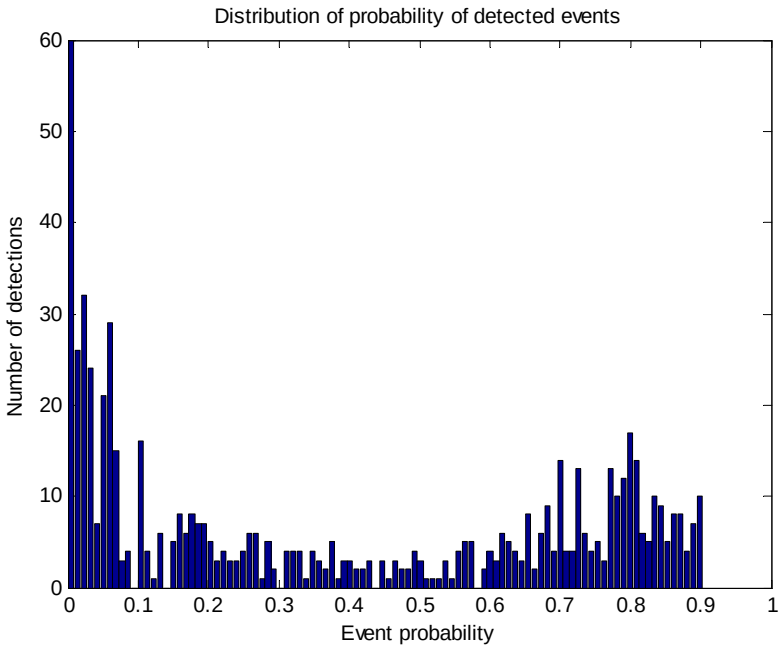


Fig. 4.10. Probability distribution of the detected events for the experiments in controlled environments. In total, 657 events were detected (filtered to 183 with a threshold of $\rho = 0.75$).

Precision and Recall of the obtained results are reported in Table 4.3. The framework presents high figures for scenarios in which foreground blobs are well detected and tracked; so the events can be easily recognized (C1 and C2). Furthermore, the included spatial constraints (*SceneContext* entity) increases the accuracy of the results by avoiding false detections in non-predefined locations. Figures notably decrease in complex scenarios (C3) mainly due to multiple occlusions, group blobs and segmentation errors (resulting in the fragmentation of foreground blobs). Additionally, non-modeled events (e.g., sitting or standing) adversely affect the detection of the modeled events. Sample results are depicted in Fig 4.11.

Cat.	LEA					GET					USE				
	TP	FP	FN	P	R	TP	FP	FN	P	R	TP	FP	FN	P	R
C1	26	2	2	.93	.93	20	2	1	.90	.95	9	0	0	1	1
C2	20	7	5	.74	.80	10	4	5	.71	.66	7	4	3	.63	.70
C3	15	16	14	.48	.51	10	7	18	.58	.35	14	10	21	.58	.40
Total	61	25	21	.70	.74	40	13	24	.75	.63	30	14	24	.68	.55

Table 4.3: Accuracy results for the analysis of controlled environments (LEA: *Leaves-object*; GET: *Gets-object*; USE: *Uses-object*).

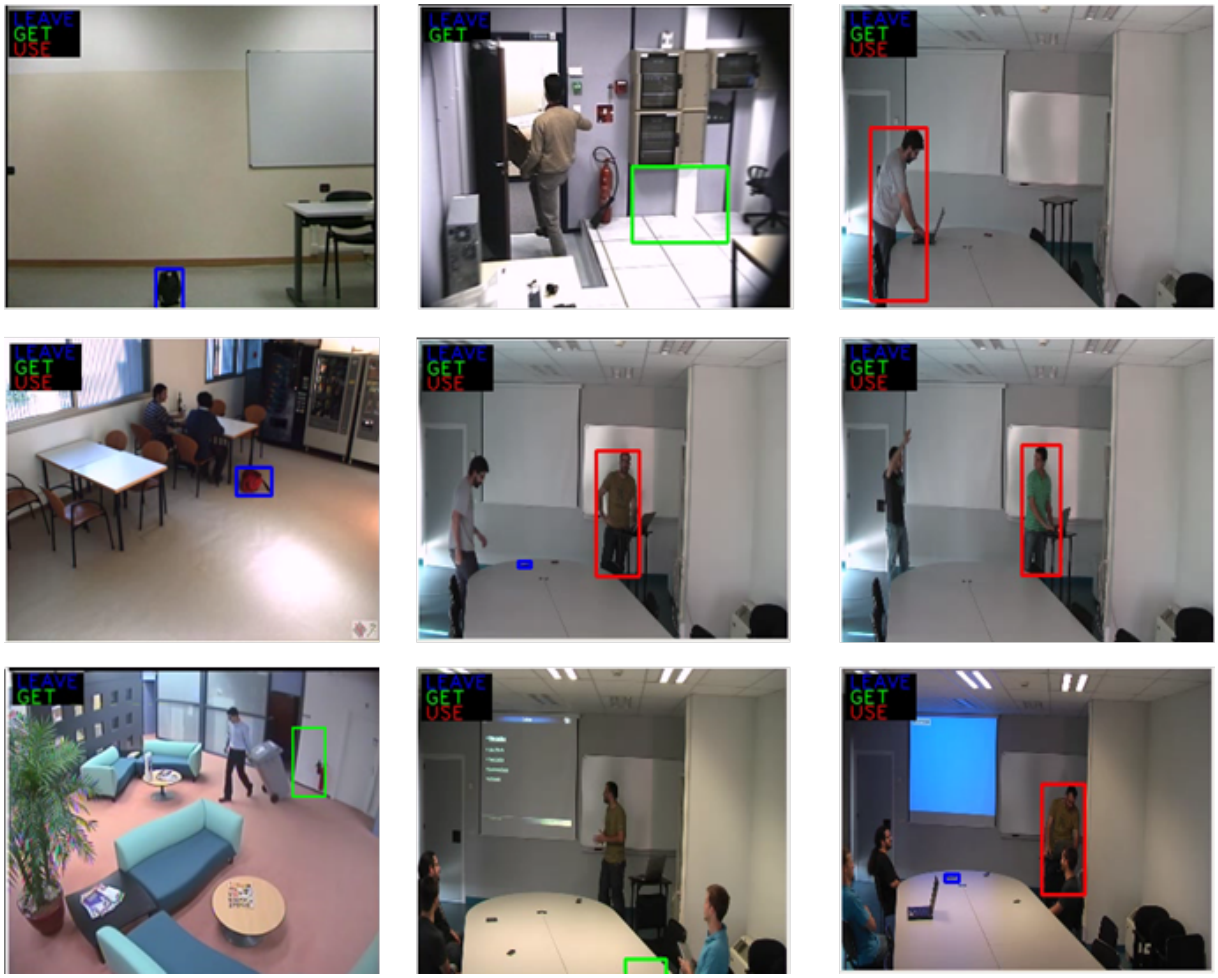


Fig. 4.11. Event detection examples for the experiments in controlled environments. Rows 1, 2 and 3 correspond to categories C1, C2 and C3. From top-left to bottom-right: *VISOR_AbandonedObject_06* (frame 213), *WCAM_indoor_activity_3* (frame 1121), *S1_0003* (frame 1974), *HERMES_cam1_indoor* (frame 616), *S2_0004* (frame 2582), *S2_0006* (frame 2737), *CANDELA_1.04* (frame 260), *S3_0001* (frame 7263) and S3_0002 (frame 5790). The color codes correspond to the *Leaves-object* (blue), *Gets-object* (green) and *Uses-object* (red).

The computational cost of the proposed approach is summarized in Table 4.4; data correspond to the average execution time for each category and stage (normalized to the size of 320x240). As it can be seen, real-time analysis is achieved with an execution time between 44.1 ms (22.6 fps) and 60.5 ms (16.6 fps) for the best and the worst cases (categories C1 and C3 respectively). The foreground segmentation stage, FG, has a (quasi) constant computational cost independently on the sequence complexity because it works at pixel-level and is blob-independent. On the contrary, blob-level analysis, BT to ED, presents a dependency on the quantity and size of blobs of interest in each sequence, being feature extraction (FE) the most time demanding stage.

Cat.	FG	BT	FE	ED	Total
C1	26.2 (59.4%)	0.5 (1.1%)	16.2 (36.7%)	1.2 (2.7%)	44.1 (100%)
C2	25.4 (47.8%)	0.9 (1.6%)	24.4 (45.9%)	2.4 (4.5%)	53.1 (100%)
C3	25.8 (42.6%)	2.1 (3.4%)	29.1 (48.1%)	3.5 (5.7%)	60.5 (100%)

Table 4.4: Average execution time for controlled environments (ms) (FG: Foreground segmentation; BT: Blob tracking; FE: Feature Extraction; ED: Event detection).

Cat.	Frames	Occurrences		Complexity			
		ABA	STO	FG	BT	FE	ED
C1	204408	54	52	L	L	M	M
C2	43632	10	4	M	M	M	M
C3	40951	15	-	H	H	H	M
C4	61951	14	-	V	V	V	V
Total	350942	93	56	-	-	-	-

Table 4.5: Dataset description for uncontrolled environments. (ABA: *Abandoned-object*; STO: *Stolen-object*; FG: Foreground seg.; BT: Blob tracking; FE: Feature Extraction; ED: Event recog.; L: Low; M: Medium; H: High; V: Very High).

4.6.3 Uncontrolled environments

For this situation, we evaluate the accuracy of the proposed approach for the recognition of the previously defined two complex events: *Abandoned-object* (ABA) and *Stolen-object* (STO).

4.6.3.1 Dataset

The evaluation dataset is composed of sequences from the CANTATA[9], HERMES[10], i-LIDS for AVSS2007[11], PETS2006[12] and PETS2007[13] public datasets. These sequences range from simple sequences with one individual to challenging sequences in crowded situations. For solving the well-known problem of background initialization [Elhabian et al., 2008], each sequence was preprocessed using a median filter to capture the initial background. In addition, a region of interest was defined for each sequence to indicate the possible location of the events[14].

Similarly to the previous experiment, we have classified the sequences into four categories attending to an estimation of the analysis complexity for each stage of the proposed framework. Table 4.5 lists the characteristics of this dataset and Fig. 4.12 shows some sample frames.

[9] http://www.multitel.be/~va/cantata/LeftObject/
[10] http://iselab.cvc.uab.es/indoor-cams
[11] http://www.avss2007.org/
[12] http://www.cvg.rdg.ac.uk/PETS2006/
[13] http://www.cvg.rdg.ac.uk/PETS2007/
[14] This was mainly done to avoid detections in highly reflective surfaces or non-interesting spatial locations.

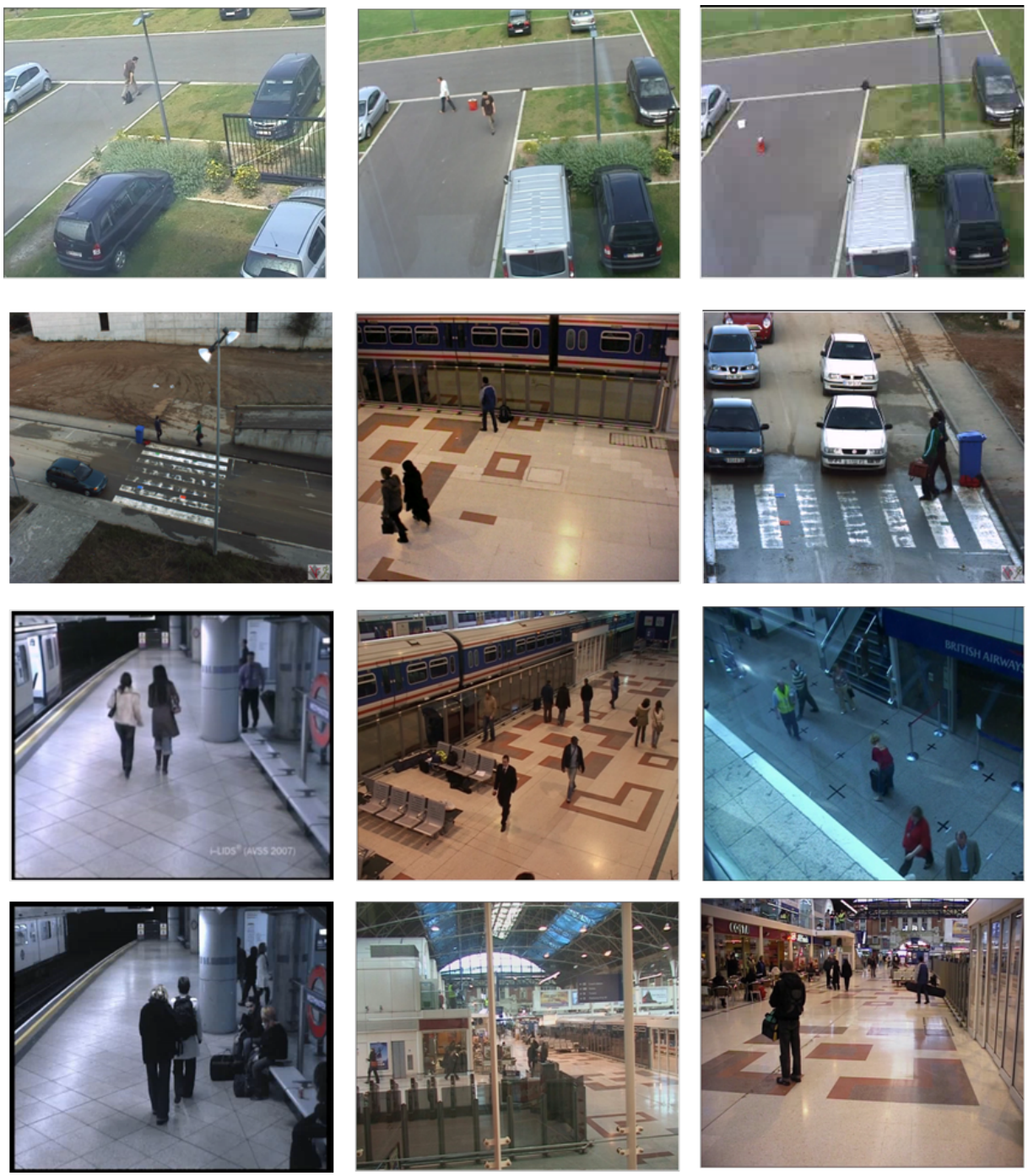


Fig. 4.12. Sample frames of the dataset for uncontrolled environments. Rows 1, 2, 3 and 4 correspond to categories C1, C2 C3 and C4. From top-left to bottom-right: *CantataMultitelCam1_018* (CANTATA), *CantataMultitelCam2_004* (CANTATA), *CantataMultitelCam2_016* (CANTATA) , *Cam2_outdoor* (HERMES), *S2-T3-C_3* (PETS2006), *Cam5_outdoor* (HERMES), *AVSS_AB_Easy* (AVSS2007), *S2-T3-C_4* (PETS2006), *S7_abandoned_bag* (PETS2007), *AVSS_AB_EVAL* (AVSS2007), *S2-T3-C_1* (PETS2006), and *S2-T3-C_2* (PETS2006).

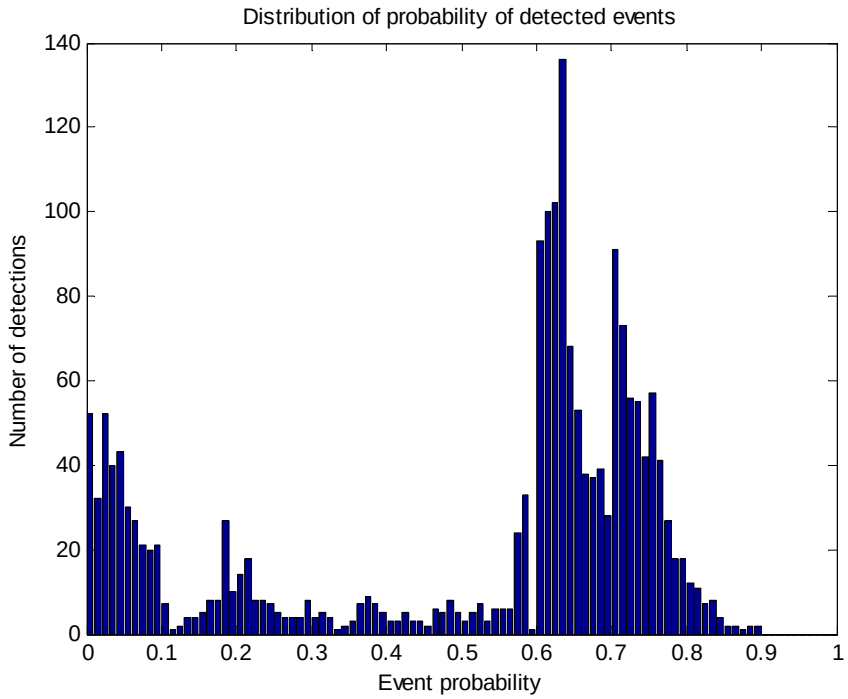


Fig. 4.13. Probability distribution of the detected events for the experiments in uncontrolled environments. In total, 2259 events were detected (filtered to 234 with a threshold of $\rho = 0.75$).

4.6.3.2 Results

In total, our approach detected 2229 event occurrences. This high number of detections can be explained by the absence of the context-based constraints (opposed to controlled environments). Fig. 4.13 shows their probability distribution. It can be observed that the event probabilities are concentrated in two ranges of values. The first one consists of events with low probability ($score < 0.1$) and they can be easily discarded as most of them are due to small segmentation errors. The second concentration is observed for intermediate-high probability ($0.6 < score < 0.8$). During experiments, it was observed that some of them were correct and some of them were due to wrong analysis of the classification modules (e.g., people recognition). Additionally, 274 events fell into an intermediate-low value ($0.2 < score < 0.6$) presenting a high uncertainty and, therefore, additional mechanisms should be used for accepting or rejecting them. After filtering their probability by using a value of $\rho = 0.75$ (see Eq. 4.12), 234 event detections were accepted.

Precision and Recall of the obtained results are reported in Table 4.6. It shows how event recognition in simple situations, such as category C1 and C2, performed reasonably well. On the contrary, the performance decreased in complex situations such as crowded scenarios. The high number of objects (moving and stationary) and the occlusions between them are the main problems that affect all the segmentation and tracking of moving objects. A high number of

Cat.	ABA					STO				
	TP	FP	FN	P	R	TP	FP	FN	P	R
C1	51	9	3	.85	.94	45	8	7	.85	.86
C2	8	6	2	.57	.80	3	6	1	.33	.75
C3	10	31	5	.24	.67	-	-	-	-	-
C4	6	20	8	.23	.42	-	-	-	-	-
Total	74	66	19	.52	.79	48	14	8	.77	.86

Table 4.6: Accuracy results for the analysis of controlled environments. (ABA: *Abandoned-object*; STO: *Stolen-object*). Note: There are no STO results for categories C3 and C4 due to the non-availability of test data.

Cat.	FG	BT	FE	ED	Total
C1	26.0 (59.4%)	0.4 (1.1%)	16.4 (37.4%)	1.0 (2.2%)	43.8 (100%)
C2	25.9 (43.6%)	1.1 (1.8%)	28.9 (48.6%)	3.5 (5.8%)	59.4 (100%)
C3	25.8 (39.4%)	2.4 (3.6%)	32.2 (49.2%)	4.8 (7.3%)	65.4 (100%)
C4	26.1 (29.4%)	6.5 (7.3%)	48.4 (54.6%)	7.4 (8.3%)	88.5 (100%)

Table 4.7: Average execution time for uncontrolled environments (ms). (FG: Foreground segmentation; BT: Blob tracking; FE: Feature Extraction; ED: Event detection).

False Positives is obtained and the *Precision* measure is decreased. A post-processing stage would be desirable to filter these detections. However, the system is able to recognize most of the events presenting acceptable *Recall* values for complex categories. Fig. 4.14 shows event recognition examples for the different categories. It should be noted the increase in the number of false positives as we analyze more complex categories.

The computational cost of the proposed approach is summarized in Table 4.7; data correspond to the average execution time for each category and stage (normalized to the size of 320x240). As it can be seen, real-time analysis is achieved with an execution time between 43.8 ms (22.8 fps) and 88.5 ms (11.3 fps) for the best and the worst cases (categories C1 and C4 respectively). Similarly to controlled environments, pixel-based analysis stages such as foreground segmentation present a (quasi) constant computational cost. The rest of the stages are blob-based and, therefore, their computational cost varies with the complexity of the sequence (e.g., the number of blobs). A notable increase of the computational cost can be observed as compared with the controlled situation (between 11-46%) due to the high density of moving objects in the scene.

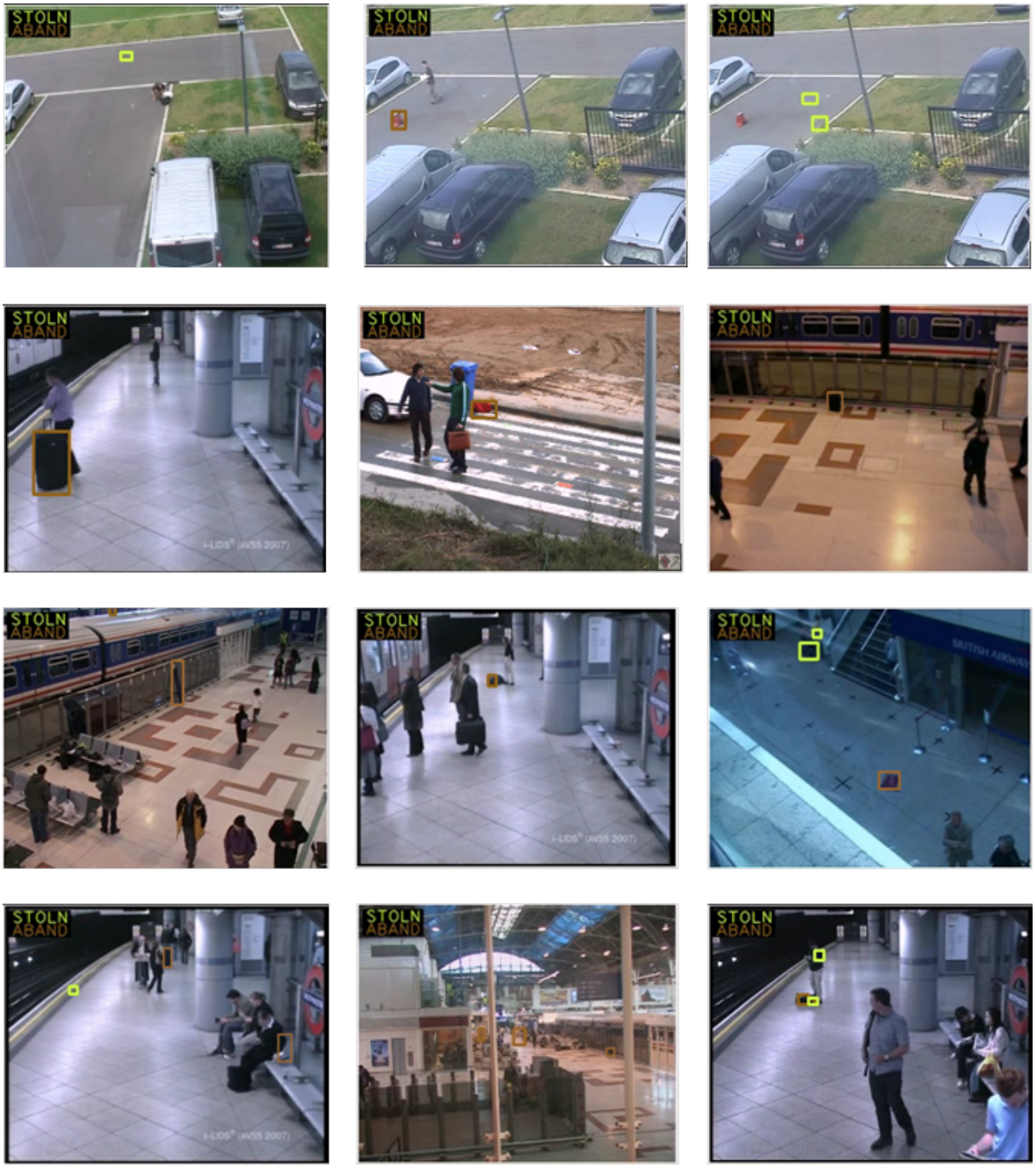


Fig. 4.14. Event detection examples for uncontrolled environments. Rows 1, 2, 3 and 4 correspond to categories C1, C2, C3 and C4. From top-left to bottom-right: *CantataMultitelCam2_018* (frame 950), *CantataMultitelCam1_013* (frame 1548), *CantataMultitelCam1_013* (frame 1745), *AVSS_AB_Easy* (frame 2451), *HERMES_Cam3_outdoor* (frame 972), *PETS06_S7_T6_B3* (frame 1641), *PETS06_S5_T1_A4* (frame 2128), *AVSS_AB_Medium* (frame 2332), *PETS07_S7* (frame 1755), *AVSS07_hard* (frame 3543), *PETS06_S6_T3_H3* (frame 2329) and *AVSS_AB_EVAL* (frame 13430). The color codes correspond to the *Abandoned-object* (brown) and *Stolen-object* (yellow).

4.7 Summary and conclusions

This chapter has described a single-view video event recognition framework guided by hierarchical event descriptions. It has been presented how the formalization of knowledge relevant to video analysis within a specific domain can be used to define strategies for event recognition. A two-layer strategy is proposed to recognize events handling the uncertainty of the low-level analysis. The *short-term* layer uses hierarchical BNs to recognize timeless events that consist of changes in object features. The *long-term* layer detects events with a temporal relation among their sub-events by means of PNs. A simple extension of the basic PN structure is proposed to manage probabilistic data related with the uncertainty of the low-level analysis. Examples for creating BN and PN detection models from event descriptions are also provided.

The accuracy of the proposed framework has been tested for the recognition of human-object interactions in controlled (short-term events) and uncontrolled environments (long-term events) for the video monitoring domain. For each situation, a heterogeneous dataset has been collected from public available datasets. For the controlled environments, a high recognition rate was achieved by exploiting the spatial relations between the moving people and the scene layout. Furthermore, a performance decrease was observed in complex situations where the accuracy and consistency of the segmentation and tracking tasks are low. For the uncontrolled environments, the results presented lower accuracy due to the non-availability of contextual information. Specially, the false positive rate was higher in complex situations because of the video data presented high complexity and variability (crowded environments). Moreover, the scene layout did not provide information to help the recognition process. In general, the precision and recall in controlled scenarios were, as expected, higher than in uncontrolled ones. Real-time operation was achieved in both situations. An in-depth study of the probability of the event detections showed that there is a high number of events with intermediate values (e.g., $0.2 < score < 0.8$). Such values can be due to uncertainty of the low-level analysis or non-modeled situations. Specifically, some problems were found related with shadow removal, background initialization, ID maintenance of tracked blobs, stability of extracted features (e.g., skin map) and people recognition (e.g., poses of moving people). Although the proposed approach is demonstrated in the video monitoring domain, it is not restricted to a specific domain knowledge or system implementation as opposed to many existing approaches.

As has been demonstrated in the work described in this chapter, the building of adequate BNs and PNs automatically from the semantic descriptions is not straightforward. The obtained results are a step ahead with respect to the existing literature, but there are still open issues for future work. For improving current recognition results, we suggest to use a feedback path for performing a top-down analysis of the events with intermediate likelihood values with the purpose of accepting or rejecting them.

Part III

Feedback-based video analysis

Chapter 5

Feedback-based analysis model

5.1 Introduction[1]

Traditionally, the processing scheme of video analysis systems is based on the *feed-forward* (or open-loop) approach that sequentially analyzes the data. The system output, computed as a function of the input data, is not used as a control variable of the processing. Its simplicity and low cost have motivated the wide spread among the existing video analysis systems. However, it does not consider the uncertainty when dealing with unexpected data and the dependence among processing levels [Nagel, 2004]. These limitations lead to low robustness of such systems for different operating conditions [Graser and Ristic, 2008].

On the other hand, the *feedback* approach is proposed as a control method to increase the robustness of the system [Franklin et al., 2009]. It defines a closed-loop control that allows to fed back the output to the input of the system. Thus, an iterative analysis is performed until a desired performance level is achieved. Despite its advantages, its application in video analysis is not very extended as the design of appropriate feedback control schemes is a complex task.

In this chapter, we present a feedback model to control video analysis. Inspired by the classical closed-loop system defined by control theory [Franklin et al., 2009], we propose a structure that allows to perform video processing tasks in a feedback operation mode. It is based on two key ideas: the analysis with different levels of detail and the complexity estimation. The former implies the analysis of the data with different degree of detail (e.g., pixel-based vs block-based segmentation) to obtain different output qualities. The latter regards the estimation of the complexity of the data to be analyzed (input) or the quality of the analysis result (output).

The remainder of this chapter is organized as follows. Section 5.2 reviews the related work. Then, section 5.3 overviews the proposed feedback model and section 5.4 discusses its implementation. Finally, section 5.5 concludes this chapter.

[1] This chapter is an extended version of parts of the publication *"J.C. SanMiguel and J.M. Martínez. Use of feedback strategies in the detection of events for video surveillance, to appear in IET Computer Vision, 2011"*

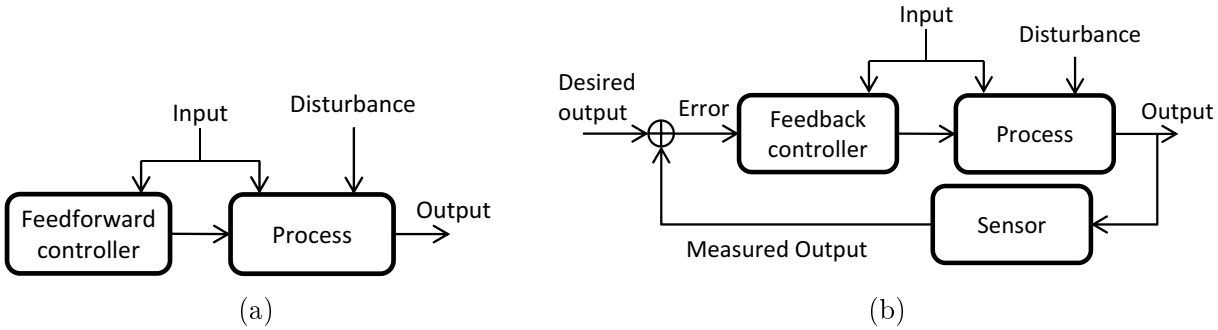


Fig. 5.1. Traditional (a) *feed-forward* and (b) *feedback* control schemes. .

5.2 Related work

As defined by control theory [Franklin et al., 2009], two broad control schemes exist: *feed-forward* and *feedback*. In this section, we review both approaches and their use in video analysis.

5.2.1 Control schemes

The *feed-forward* (or open-loop) approach groups systems in which their output has not effect on their input at the same time instant. In such systems, two basic components are predominant (shown in Fig. 5.1(a)): the process and the controller. The former defines the operations carried out to fulfill the system purpose and the latter describes the management of the process. This approach refers to systems where the communication between these two elements is one way. A direct path from input to output exists and the controller does not know if the process is operating as expected. However, real-world applications may present some external disturbances (e.g., low-quality input data) that affect the system performance. Hence, the feed-forward controller may not be adequate as it tries to predict and compensate the effect of these disturbances for obtaining the desired result.

In contrast, the *feedback* (or closed-loop) approach defines systems in which their output has effect on its input at the same time instant. The output (or part of it) is measured and compared against a reference value to estimate its deviation (error signal). Then, this signal is fed back into the system input in order to minimize this deviation. This approach enhances the feed-forward scheme by including a module for the measure task (sensor) and a reverse path to influence the feedback controller as depicted in Fig. 5.1(b). Hence, it does not have to predict the effect of disturbances in the system output (unlike the feed-forward approach) because it measures their impact and alters the system input to reduce the deviation from the desired output. Thus, the main advantage of the feedback approach is its ability to track the system performance. However, the design complexity is highly increased as compared with the feed-forward approach. A comparative analysis of both approaches is given in Table 5.1.

Feed-forward (open-loop)	Feedback (closed-loop)
Advantages • Simplicity (fewer components are needed). • Low-cost (instantaneous time response).	Advantages • Increases robustness to unpredicted conditions. • Ensures a desired performance level of output.
Disadvantages • No tolerance to disturbances or unforeseen data. • Does not know if the system is performing as expected.	Disadvantages • High design complexity (feedback controller and sensor). • Additional time cost (delay due to the iterative process).

Table 5.1: Comparative analysis between the main approaches for system control.

5.2.2 Feedback-based video analysis

Most of the current video analysis literature is based on the *feed-forward* approach that sequentially processes the input data. Moreover, the analysis modules are divided into low-level and high-level processing depending whether they focus on raw image data or higher levels of abstraction. In this sequential processing chain, it is well-known that the existing dependence of the high-level results with respect to the accuracy of the low-level processing [Nagel, 2004] (e.g., a poor segmentation makes recognition very difficult). Furthermore, low-level processing is expected to operate in challenging situations (e.g., image noise, varying illumination conditions) that may affect its performance requiring to use different parameter configurations or low-level techniques for achieving the optimum accuracy. Therefore, low robustness is expected when the feed-forward approach is applied to real-world conditions [Graser and Ristic, 2008]. In this situation, it would be desirable to introduce feedback mechanisms for sending information from the higher to the lower processing levels to improve the system performance.

For applying *feedback* in video analysis, it is necessary to model four key elements: the control variables, the reference, the measures and a set of rules. Firstly, control variables are required to directly influence the analysis process (e.g., the algorithm parameters). Secondly, a reference output is needed for establishing a desired output quality level. This reference can be determined using ground-truth data (e.g., the output that the system has to ideally achieve) or without it (e.g., reduce the uncertainty of the output data). Thirdly, measures of the obtained output have to be designed for reflecting the output quality with respect to the desired level (e.g., ground-truth evaluation). Finally, a set of rules have to be devised for modeling the change of the control

variables (e.g., increase the parameter value when a decrease in quality is observed). As it can noticed, the use of feedback in video analysis is a complex task. This fact confirms the widespread adoption of the feed-forward approach as its design is simpler than the feedback one. Among existing feedback-based approaches for video analysis, we differentiate between those based on supervised or automatic feedback.

5.2.2.1 Supervised feedback

Supervised feedback-based approaches use external information (usually given by the user) for evaluating the video analysis task. Two categories exist based on how this feedback is provided.

The first one describes an explicit inclusion of the user feedback in the video analysis loop. Based on the *Relevance feedback* concept [Rui et al., 1998], it is proposed to have a system monitored by a user who judges the quality of the result. Then, some rules are defined to automatically translate this feedback into control variables of the system. For example, [Shekhar et al., 1999] defined a vision system that employed output quality indications, given by the user (high-level feedback), for self-tuning. Moreover, [Oerlemans and Thomee, 2007] presented a tracking system that includes user feedback for detecting moving regions. The user provides information at pixel (e.g., wrong detections) and object (e.g., feature accuracy) level.

The second category describes the implicit usage of user feedback. Commonly, ground-truth information (the reference) is generated by the user and provided to the system for achieving optimum performance by means of parameter adjustment [Bins et al., 2005] and model learning [Hall, 2006]. More recently, [Sherrah, 2010] proposed to use this reference to learn the relations between the input and output of the system in a training phase. Additional features are included to measure the input and the output performance. Its novelty relies on the learning of incremental input-output changes instead of the absolute changes (i.e., ground-truth evaluation).

Despite both categories improve results, they are limited for real-time operation (as the user feedback is a high time-consuming process) or real-world conditions (as they use labeled data).

5.2.2.2 Automatic feedback

There are few works addressing the automatic use of feedback in video analysis. Some of them describe a generic model that can be applied in different levels of the video analysis. For example, [Mirmehdi et al., 1999] presented how the traditional feedback approach is applied for object recognition. Cost functions associated to the processing levels are defined with the objective of determining the optimal features to use and searching for missing information. Finally, these feedback strategies are combined to get the highest accuracy. Moreover, [Harville, 2002] adapted the concept of positive and negative feedback to incorporate corrective guidance from the high-level to the low-level processing. It is demonstrated in the video surveillance domain where object detection modules are used to feedback a foreground segmentation stage. Moreover, [Graser and

Ristic, 2008] defined a basic feedback structure composed of a sensor and a control variable. Then, it is applied to the image segmentation task where the sensor measures the connectivity of segmentation masks and a particular parameter of the algorithm is used as the control variable. In [Hotz et al., 2008], a generic high-level interpretation system is described in which feedback is used to generate hypothesis for guiding the low-level processing. Feedback is interpreted as a coarse-to-fine strategy that allows to focus the analysis effort where it is required. In [Garcia and Bescos, 2008], a knowledge-based framework is presented based on the generation of confidence maps in each analysis level and a posterior verification of these maps in the superior analysis levels using them to feedback the low level stages. This approach is validated for foreground segmentation by combining the frame difference and background subtraction methods.

Regarding the specific application of feedback without using a structured model, most of the proposals are focused on the moving object segmentation and tracking stages. For example, [Cavallaro and Ebrahimi, 2004] detects video objects based on the interaction of region-based and motion-based object segmentation stages. It can be seen as a type of feedback where, for each frame, the motion analysis provides the focus of attention of the objects to detect and it is improved by the feedback from the region analysis of these objects. Moreover, [O'Conaire et al., 2006] adaptively computed thresholds for foreground detection maximizing the mutual information between foreground maps calculated for visual and thermal infrared images (two detectors). The output of each detector is used to feedback the other one for adjusting its parameters to maximize the agreement between both detectors. Similarly, the same scheme is applied for shadow detection [O'Conaire et al., 2007b]. In [Rincon et al., 2007], a feedback model is presented to handle segmentation errors for people detection. Specially, the person model uses a decomposition strategy in description levels to enable the feedback of information between adjacent levels translating it into a parameter configuration that affects the segmentation process. In addition, [Wang et al., 2009] extended the previous approaches by proposing a coupled detection-tracking system based on shape, color and velocity information. Relations between both stages are studied to restrict the parameters of the detection stages and remove wrong detections due to noise. The ratio between the 2-D projections of the target shape is used as a confidence measure of this system. Using a similar idea, [Torabi et al., 2010] proposed to feedback the segmentation module with tracking data using thermal and color data. Specific feedback-based tracking analysis is discussed in [Erdem et al., 2004b] where a contour tracking algorithm is fed back by confidence measures that study the variability of the target appearance and the contour contrast (in terms of color and motion).

For other processing tasks, not many feedback-based works exist. For instance, [Zhou et al., 2006] used feedback for self-adapting object detection and recognition in varying illumination conditions. This problem is modeled as an optimization using the quality of the recognized object as the goal function and the feedback of information from the recognition to the detection

is used to adapt the applied illumination model. For event detection, [Song et al., 2007] defined a framework for on-line selecting the most suitable event model according to tracking data. An automatic evaluation of tracking performance is fed back to the event recognition module to switch the recognition model. Moreover, [Carmona et al., 2009] corrected low-level analysis using specific routines derived from expert knowledge for successfully recognizing events.

5.2.3 Conclusion

Feedback strategies have been mainly used in video analysis to improve the results of individual processing stages as compared to similar approaches without using feedback. Current literature exhibits two main problems for implementing feedback strategies at system and single processing stage levels. From the system viewpoint, the modeling of the relations between high, mid and low-level analysis stages is a very complex task. This fact has motivated that most of the approaches are focused on the low-level stages using simple feedback loops for corrective guidance. From the processing stage viewpoint, a difficulty is encountered for evaluating the quality of its output in real-world conditions (i.e., no available ground-truth data) and modifying the response of the analysis system (i.e., implementing the feedback controller).

5.3 Proposed model for feedback-based video analysis

We propose a feedback processing scheme to solve the limitations of the feed-forward approach in video analysis. Its application on individual processing stages aims to enhance their results independently of the performance of other system stages. It can also be introduced in the stages of a system to improve its final results, although in some stages there could be a reduction in their performance. In this situation, a mechanism is required for coordinating the feedback-based stages. This section overviews the key concepts and the structure of this feedback model.

5.3.1 Key concepts

The proposed feedback model is based on two key concepts: levels of detail for the analysis and complexity estimation.

The *Level of Detail* concept (LoD) represents the analysis of the input data with different configurations that modify the behavior of the analysis system and impact on the quality of its output. The LoD configurations correspond to variations of system parameters, optimum algorithm selection, algorithm combination and transformations of the input data (a further discussion is provided in subsection 5.4.1). The levels of detail have to be properly defined and sorted considering its performance. It is assumed that an analysis with higher LoD will produce output results with higher or equal quality but never with less quality. Fig. 5.2 shows an example for foreground segmentation with different LoD for the analysis.

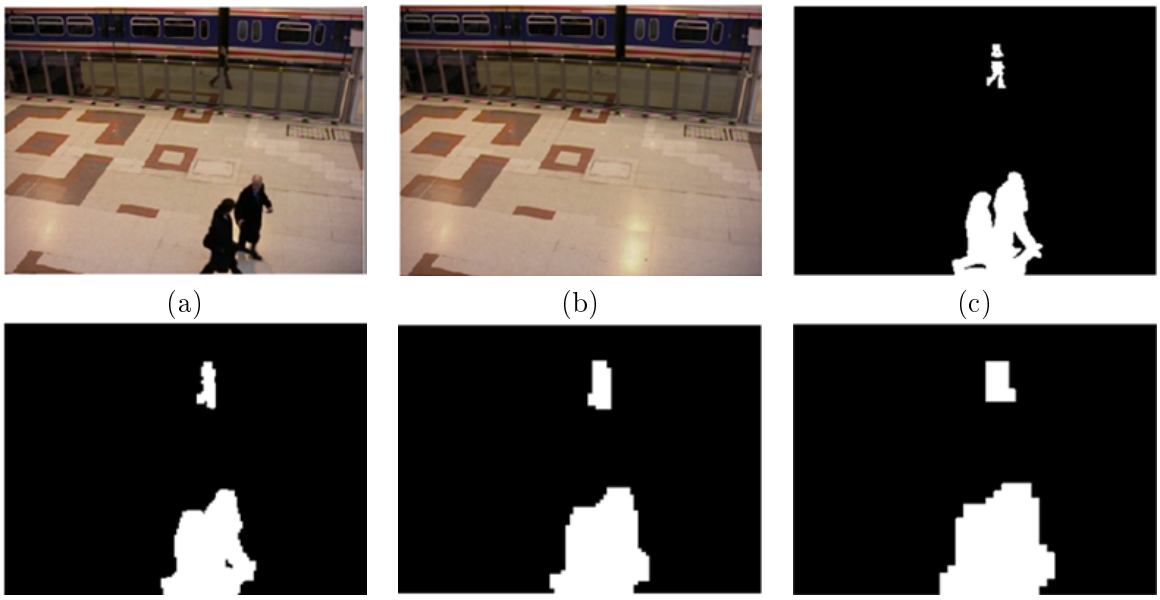


Fig. 5.2. Foreground segmentation results with different level of detail (LoD). Data corresponds to (a) current frame, (b) estimated background at pixel resolution and foreground segmentation maps using [Cavallaro et al., 2005] with (c) 1x1, (d) 2x2, (e) 4x4 and (f) 8x8 pixel blocks.

Fig. 5.3. Sample frames with different complexity for their analysis. Data correspond to (a) VISOR, (b) CVSG and (c) AVSS2007 datasets.

The *Complexity Estimation* concept (CE) regards the evaluation of the data to be analyzed (input) or the analysis result (output). Similarly to the prediction of the *feed-forward* control scheme, the estimation of the difficulty of the input data could be useful for determining which LoD (e.g., configuration) is the optimum one to use. Therefore, more analysis effort will be employed if the data presents high complexity. An example of the different input complexities is shown in Fig. 5.3. Additionally, the evaluation of the output quality gives information to the feedback scheme for achieving the reference output level by means of changing the LoD. Fig. 5.4 illustrates the application of different foreground segmentation approaches. In this case, a

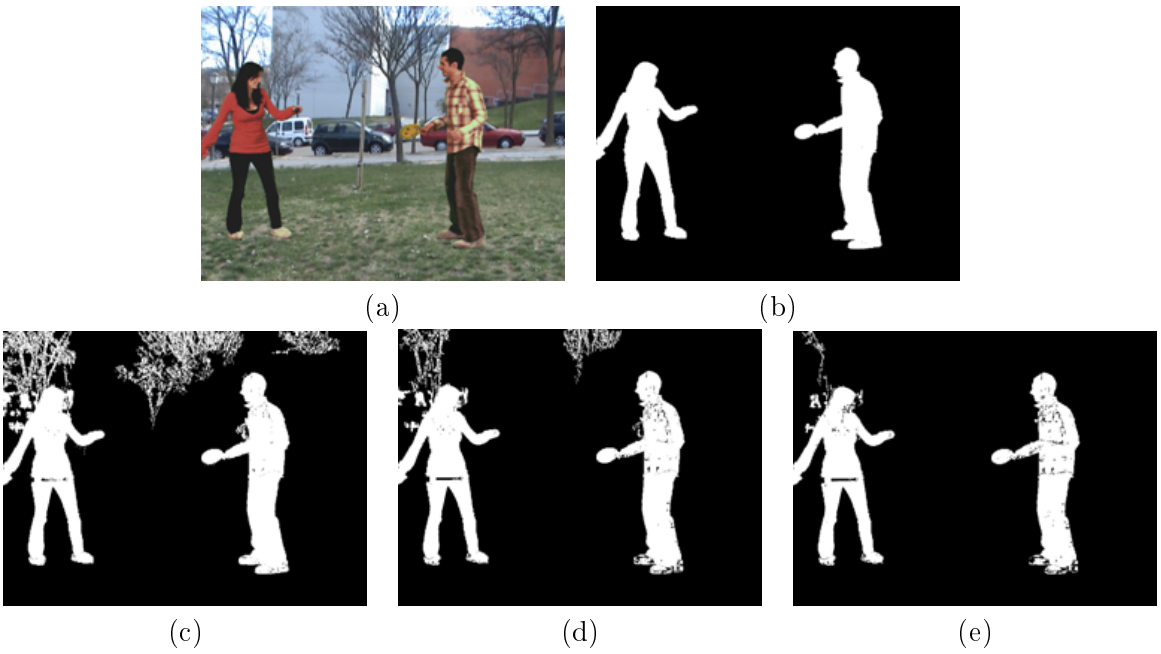


Fig. 5.4. Sample results for different foreground segmentation methods. Data corresponds to (a) current frame, (b) ground-truth and results using the (c) MoG [Stauffer and Grimson, 1999], (d) KDE [Elgammal et al., 2000] and (e) EigBKG [Oliver et al., 2000] approaches.

method to estimate their output quality (complexity) is required to choose the optimum LoD. A further discussion for implementing the complexity estimation is given in subsection 5.4.2.

5.3.2 Structure

Inspired by the closed-loop systems defined by control theory [Franklin et al., 2009], we propose a model based on the previously mentioned concepts. Its structure has three components (depicted in Fig. 5.5): the *processing stage*, the *estimator* and the *actuator*.

The *processing stage* is the component that performs the analysis. In order to allow variable analysis effort, a number of levels of detail (LoD) is predefined between 1 and *MAXLoD* (MAXimum Level of Detail) that corresponds to the followed strategies for implementing the LoD (e.g., different parameter settings). Thus, the output is generated as follows:

$$Output(t) = P\big(Input(t),\ LoD(t)\big), \tag{5.1}$$

where *Input* represents the data needed for performing the task and *LoD* refers to the current *Level of Detail* selected by the *actuator* (that changes for the same instance in order to reach the desired performance level).

The *estimator* measures the difficulty of the data to be analyzed and the quality of the output results. Since the ideal complexity measures are either non-computable or infeasible for practical

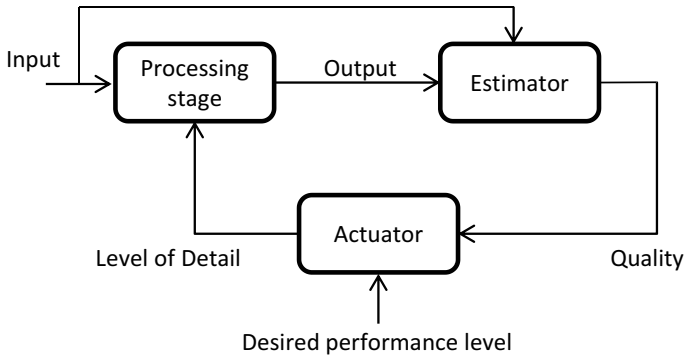


Fig. 5.5. Structure of the proposed feedback model for video analysis.

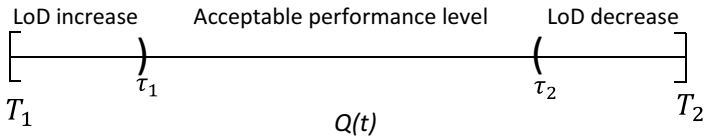


Fig. 5.6. Acceptable performance level for each Level of Detail (LoD)

applications, qualitative estimators have to be proposed depending on each considered *processing stage*. For the input complexity, a function F_I is defined to study the variation of the data in the analyzed feature spaces by the *processing stage*. For estimating the output quality, the function F_O inspects properties of the output data (and its variation along time). For each iteration of the feedback loop at time t, both functions are combined as follows:

$$Q(t) = \omega F_I\big(Input(t)\big) + (1-\omega)F_O\big(Output(t)\big), \tag{5.2}$$

where $\omega \in [0,1]$ is the weighting factor of the combination.

The *actuator* decides which level of detail is selected for the analysis of the *processing stage*. It implements the rules to control the behavior of the analysis. As a first approach, this decision is based on defining thresholds for increasing and decreasing the LoD as follows:

$$LoD_i(t) = \begin{cases} LoD_{i-1}(t)+1 & Q_{i-1}(t) < \tau_1 \\ LoD_{i-1}(t)-1 & Q_{i-1}(t) > \tau_2 \\ LoD_{i-1}(t) & otherwise, \end{cases} \tag{5.3}$$

where $LoD_i(t)$ is the LoD at time t and iteration i; $Q_i(t) \in [T_1, T_2]$ $(T_1 \neq T_2)$ is the complexity estimation at time t and iteration i; and τ_1 and τ_2 are the thresholds for changing the LoD. Implicitly, these thresholds define a zone of acceptable performance level where there is no increase or decrease of LoDs (see Fig. 5.6) and the iterative process is finished.

5.4 Approaches for implementing the feedback model

A number of existing approaches can be used as components of the proposed feedback model. In this section, we identify them and describe how they can be applied.

5.4.1 Level of detail

In the literature, there exist analysis strategies that can be easily incorporated to implement the LoD concept. As a first approach, we have selected four well-known approaches: *incremental focus of attention* [Toyama and Hager, 1999], *iterative data analysis* [Salembier and Ruiz, 2002], *multi-resolution analysis based on Regions Of Interest (ROI)* [Lefevre et al., 2002] and *variable-configuration* [O'Conaire et al., 2007b].

- The *incremental focus of attention* approach [Toyama and Hager, 1999] proposes a layered hierarchy of available algorithms to perform a task. All layers are sorted by output precision (usually this means that faster/slower algorithms will tend to occur at the bottom/top layers). The processing capabilities are determined by the selection of a particular algorithm for the task. Consequently, the LoD can be defined as the use of a specific layer or algorithm.

- The *iterative data analysis* approach [Salembier and Ruiz, 2002] exploits the iterative data analysis performed by some algorithms. This process usually improves the result of the analysis with each new iteration and it is stopped by some predefined criteria. Thus, the LoDs of this type of approaches can be defined as a modification of the stopping criteria and the parameters involved in the iterative process. No specific LoD sorting is required if the analysis process always improves results with each new iteration.

- The *ROI-based multi-resolution* approach [Lefevre et al., 2002] transforms the input data for its processing at different resolutions. The ROIs are used to guide the analysis process and they are obtained in the lowest resolution level of the analysis. Then, they are refined with the analysis at higher resolutions. This approach can be considered as a coarse-to-fine strategy. It is assumed that higher resolutions produce more accurate results. Hence, the LoDs can be defined as the number of input resolution modifications.

- The *variable-configuration* approach [O'Conaire et al., 2007b] relies on having various system configurations (e.g., parameter settings) to produce different outputs. A measure to estimate the output quality needs to be defined. Then, the system configurations are selected following an optimum search strategy guided by this measure. Therefore, the LoDs can be defined as the available system configurations. They have to be appropriately sorted to assure that higher LoDs produce more accurate results (i.e., better quality).

5.4.2 Complexity estimation

Currently, there are attempts to quantify the complexity of the input data as well as the output quality for a particular video processing stage. Although they have shown a relative effectiveness in short or low-complexity videos, they are application dependent in most of the cases and they have not been tested in complex datasets making difficult to extrapolate the results of the complexity estimation. For estimating input complexity, we differentiate the following approaches:

- *Identify the associated context* to the task. For example, a foreground segmentation algorithm could benefit from a prior analysis of the scene background to identify homogeneous background regions for obtaining theirs optimum parameters (e.g., image noise). Moreover, different tracking algorithms are applied depending on the distance between the tracked objects [Tyagi and Davis, 2008].

- *Use additional features.* External features can be employed for studying properties of the input data that may affect the analysis performance. For instance, [Sherrah, 2010] proposed to use the number of bright pixels in the input image to increase or decrease the threshold of a foreground segmentation algorithm.

For estimating the output quality, we differentiate the following approaches:

- *Use ground-truth information.* Despite its drawbacks for real-world applications, it is still applied to evaluate the output quality using labeled data and to feedback the algorithms such as self-tuning of color segmentation [Georis et al., 2004] and tracking [Hall, 2006].

- *Use additional features.* Complementary features to the ones used in the processing stage can be identified to measure quality. For example, [Erdem et al., 2004a] used the feature contrast (in terms of color and motion) for evaluating segmentation and tracking data.

- *Temporal coherency of results.* The correlation of the results at different time instances can be used to estimate the output quality allowing to detect unexpected results. For example, the particle filter based object tracking provides measures of the output data likelihood [Nummiaro et al., 2003]. In this case, the track over time of this likelihood could be used to detect performance drops [Vaswani, 2007].

- *Use reliability measures.* The objective of this approach is to test the consistency of a set of measures obtained from the output data. For example, [Maggio et al., 2007] computed the spatial uncertainty of a multi-hypothesis tracking algorithm to determine the reliability of its tracking data.

- *Consistency relative to a priori model*s. Based on the availability of object models, it estimates the output quality as the result of the model fitting process. For example, [Harville,

2002] measured the quality of foreground segmentation data with different high-level detection modules for the expected objects (people, non-people and illumination changes).

- *Agreement between independent algorithms*. For tasks where a set of independent algorithms is available, it is possible to study their output similarity for measuring the output quality, for example, for foreground analysis using the infra-red and visible spectra [O'Conaire et al., 2006] and HSV-based shadow analysis [O'Conaire et al., 2007b].

5.5 Summary and conclusions

In this chapter, we have presented the benefits of the *feedback* approach for video analysis compared with the traditional *feed-forward* one. Its main advantages are the possibility of achieving a desired performance level and dealing with unexpected conditions. However, its complex design has limited its use by the video processing community. Conversely, the *feed-forward* approach is easy to develop and the control over the process performance is not feasible.

For solving some of existing limitations, we have proposed a *feedback* model for video analysis. It is based on having variable levels of detail for analysis and the complexity measurement of the data to be analyzed as well as the output of each processing stage. This model can be applied to manage the processing effort (i.e., computational cost) of the system maintaining a desired performance level. Furthermore, existing approaches that can be applied to the components of the model have been identified and discussed.

In the following chapters, we study how the proposed feedback model can be inserted into video analysis applications or single processing stages to enhance its robustness.

Firstly, we have proposed its use in a typical video surveillance system oriented to event recognition (chapter 6). This feedback model is introduced into the foreground segmentation, shadow removal, people recognition and event detection stages of such system. These feedback-based stages are coordinated with the objective of improving the system final results (e.g., event detection, computational cost) and not necessarily increasing the performance of each stage.

Secondly, it was observed that the main difficulty for using feedback consists in the estimation of the output quality for each processing stage. Hence, we have decided to study the estimation of this quality without using ground-truth information (i.e., self-evaluation) for the most common stages of existing video analysis systems: foreground segmentation and object tracking. This study can be considered as the previous phase for applying the feedback model to individual processing stages in real conditions when ground-truth information or user feedback are not available. We have conducted a study of the related literature and compared the most representative approaches (chapter 7). Finally, we have proposed a novel method for self-evaluation of video object tracking approaches to solve the limitations of current literature (chapter 8).

Chapter 6

Feedback-based event recognition

6.1 Introduction[1]

Video event recognition is a key analysis stage in semantic video content analysis. Most of the previous research in the area considers as independent all the processing stages that compose an event detection system. However, recent works have demonstrated a high relation between processing stages (e.g., segmentation, tracking, recognition) that affects the accuracy of the event recognition task [Nagel, 2004; Carmona et al., 2009].

In this chapter, we present a feedback-based approach to recognize events in video. The feedback model proposed in chapter 5 is applied to a traditional event recognition system (hereafter called base system) to improve its performance in terms of accuracy and computational cost. The estimation of the data complexity is used to adjust the computation effort of the analysis stages with the objective of improving the system performance (e.g., maintain accuracy whilst reducing computational cost). Three feedback strategies are described and introduced in the processing stages of this system. For managing the interaction between these strategies, a rule-based system is designed to estimate how high-level analysis is affected by the level of detail change of the processing stages. Finally, the benefits of the proposed approach are demonstrated over a dataset composed of sequences with varying complexity.

This chapter is structured as follows. Firstly, section 6.2 briefly describes the base system that generates the event detections to introduce the feedback strategies. Then, section 6.3 overviews the feedback-based system. After that, section 6.4 describes how the feedback strategies are applied and section 6.5 the mechanism to coordinate them. A summary of the achieved results is presented in section 6.6. Finally, section 6.7 summarizes this chapter with some conclusions.

[1] This chapter is based on the publications "*J.C. SanMiguel and J.M. Martínez., Shadow detection in video surveillance by maximizing the agreement between independent detectors, in Proc. of IEEE Int. Conf. on Image Processing, pp. 1141-1444, Cairo (Egypt), Nov. 2009*" and part of "*J.C. SanMiguel and J.M. Martínez. Use of feedback strategies in the detection of events for video surveillance, to appear in IET Computer Vision, 2011*"

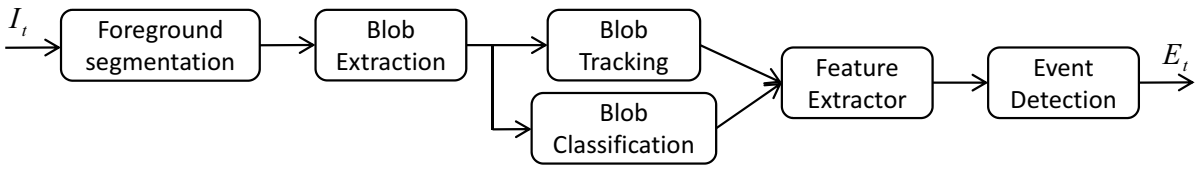


Fig. 6.1. Block diagram of the base system for video event recognition

6.2 Base event recognition system

We present a framework for recognizing human-related events that will allow us to evaluate and compare the performance of the feedback strategies to be applied. It has been set up with state-of-art techniques for each processing stage. This system operates as part of a real-time video-based surveillance system. It implies that the computational complexity of the processing algorithms should not be high. Hence, the decisions on selecting the analysis algorithms are affected by their computational cost as well as quality of their outputs. In this section, we review its structure and the event recognition module.

6.2.1 Analysis modules

The architecture of the base video analysis framework is depicted in Fig. 6.1. After a frame acquisition stage, several analysis modules are sequentially applied:

- *Foreground segmentation.* It detects the moving objects based on adaptive background subtraction and statistical change detection [Cavallaro et al., 2005]. Then, foreground data is filtered by a shadow removal stage based on the HSV color space [Prati et al., 2003].
- *Blob Extraction.* It groups the foreground pixels into connected regions by using the two-pass connected component analysis [Szeliski, 2011] and extracts their bounding boxes.
- *Blob Tracking.* It associates the detected blobs. A Kalman filter predicts the position for the blobs of the previous frame and, then, it computes the spatial and color distances with the blobs of the current frame to get their real position [Caporossi et al., 2004].
- *Blob classification.* The likelihood (score) of being people is computed for each blob based on the combination of geometrical bounding box analysis, ellipse adjustment and location of the head/shoulder contour [Fernandez-Carbajales et al., 2008].
- *Feature Extraction.* It calculates some features (e.g., speed, direction, mean color) for the blobs under analysis in the event detection stage.
- *Event Recognition.* It detects events using the information from the previous stages. This stage is described with more detail in subsection 6.2.2.

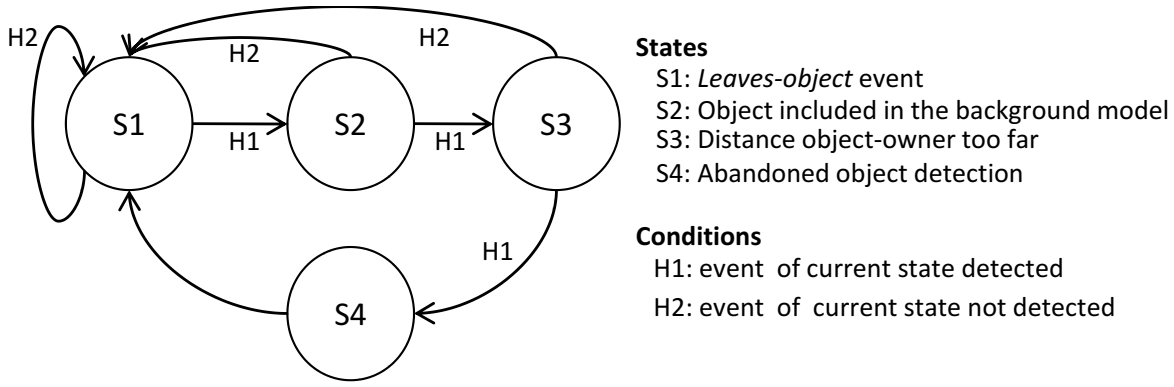


Fig. 6.2. Finite-State-Machine model for the complex event *Abandoned-object*.

6.2.2 Event modeling

Similarly to chapter 4, we recognize two simple events (*Leave-object* and *Get-object*) and two complex events (*Abandoned-object* and *Stolen-object*). In particular, we use the models for uncontrolled conditions defined in subsection 4.5.3. Furthermore, we avoid the strategy of modeling the events related to the recognition problems (defined by using expert knowledge [Martinez-Tomas et al., 2008]). Therefore, the recognition of complex events can be simplified to use a Finite-State-Machine (FSM) where each state corresponds to *simple* events as in [Hongeng et al., 2004]. For modeling the transitions between states in each FSM, we define the hypothesis H_1 and H_2 for, respectively, moving to the next and to the initial state. Fig. 6.2 shows the definition of the *Abandoned-object* event.

6.3 Overview of the feedback-based system

The feedback model of chapter 5 is introduced in the processing stages of the base system to improve the event detection accuracy and to adjust the computational cost of the stages to the complexity of the situation. Specifically, the event detection improvement is achieved by re-inspection of events detected with high uncertainty, named *unknown* events (in our case, events detected with intermediate likelihood). An increase of the *Level of Detail* (LoD) is done for the processing stages operating at a low performance level. The adjustment is focused on the computational cost adaptation for simple and complex situations by, respectively, reducing or increasing the LoD of the processing stages. A rule-based system manager is designed to handle the feedback strategies. It is in charge of recollecting the input and output data, of evaluating the performance of the processing stages and of generating the corresponding feedback signals for LoD change. Hence, it acts as the controller of the feedback model. Fig. 6.3 depicts the block diagram of the feedback-based analysis system. The application of the feedback model to each processing stage and their coordination are detailed in the following sections.

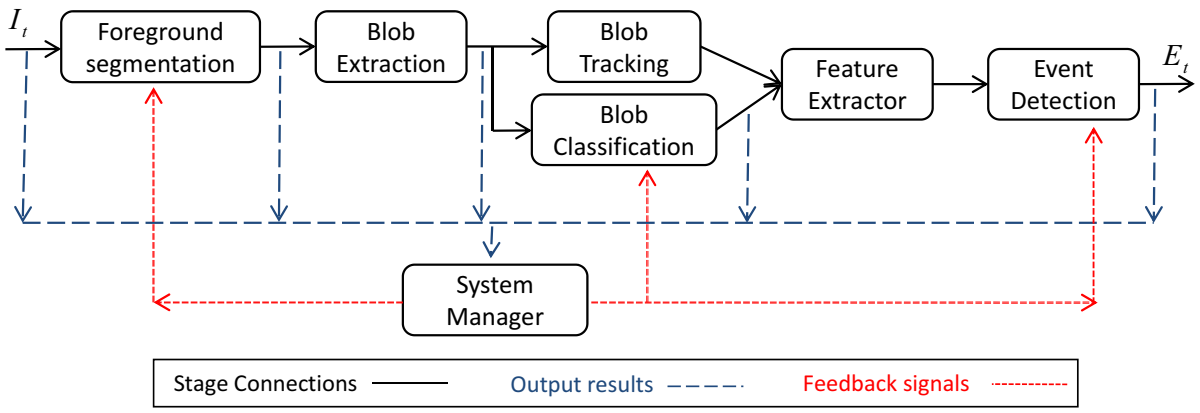


Fig. 6.3. Feedback-based video event recognition system. Solid, dotted and dashed lines indicate connections, respectively, between analysis stages, from the analysis stages to the system manager and feedback signals to the analysis stages.

6.4 Feedback implementation in the processing stages

The feedback model is implemented in the following processing stages of the system: Foreground segmentation (background subtraction and shadow removal), Blob Classification and Event Detection. This section describes how the feedback model is applied.

6.4.1 Feedback-based background subtraction

For the background subtraction task, the *ROI-based multi-resolution analysis* approach has been selected (see section 5.4.1). As segmentation methods usually rely on analyzing pixels or regions, multi-resolution analysis is appropriate for obtaining outputs with different qualities. We propose to combine the regular pyramidal decomposition [Lefevre et al., 2002] with the background subtraction approach defined in [Cavallaro et al., 2005].

6.4.1.1 Levels of Detail (LoDs)

The LoDs correspond to the levels of the pyramid structure (i.e., the different resolutions). For computing the pyramid structure, we propose to create such multi-resolution representation of the image using a regular Gaussian pyramidal representation (see Fig. 6.4). In this structure, inter-level relationships are fixed so the structure only reduces the resolution of the input image in the consecutive levels. From an original image (n_0), each pyramid level n_i is recursively obtained by processing its underlying level n_{i-1}. Specifically, the image at level n_i is convolved with a Gaussian filter (5x5) and then down-sampled by rejecting even rows and columns. Thus, there is a reduction by two in width and height. Finally, we iterate to obtain the lowest resolution image of the pyramid that corresponds to n_{max}.

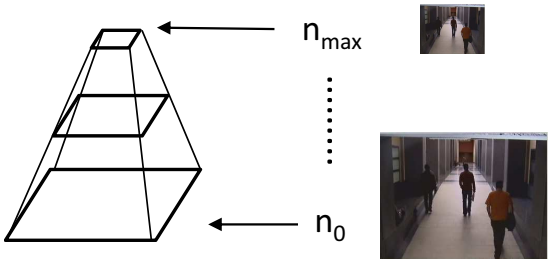


Fig. 6.4. Pyramidal representation of the feedback-based background subtraction method. Levels n_0 and n_{max} describe, respectively, the original and minimum frame resolutions.

We assume the fact that when we look at an image from a far viewpoint we mainly see the background image. Thus, the segmentation process starts at the pyramid level n_{max}. The calculated segmentation results are propagated towards the lower levels of the pyramid. If we want to grow from level n_2 to level n_1, the segmentation of level n_2 can be propagated to level n_1 by applying three stages. Firstly, we have to calculate the initial foreground mask (*mask1*) for level n_1, this image is calculated by up-sampling the corresponding image in the previous level n_2 (*mask2*). Secondly, a foreground pixel analysis (in *mask1*) is performed to calculate the bounding boxes of these pixels. Finally, these detected bounding boxes (ROIs) are refined by applying the same background subtraction process as the one used in level n_0 to obtain the foreground mask of level n_1. In order to avoid that the upper background models of the pyramid become in a non-updated state (the foreground seed, level n_0, is always processed whereas the other LoDs are obtained depending on the desired level); each N frames the whole pyramid model is updated. A description of the iterative process for choosing a LoD and obtaining the desired foreground mask is given in Algorithm 6.1.

6.4.1.2 Complexity estimation

As complexity estimator, we use a heuristic based on the foreground percentage with respect to the image size trying to measure crowded situations that usually lead to environments more difficult to analyze. A running average scheme is used to accumulate its values. For example, if we detect a low motion percentage in the last N frames, we can decide to reduce the LoD of the segmentation module. On the other hand, if the sequence becomes crowded, the system must increase the LoD to refine the detection allowing the identification of the individuals in later analysis stages. This complexity at time t, $Q_{FG}(t)$, is estimated as follows:

$$Q_{FG}(t) = (1 - \alpha)FGP(t) + \alpha FGP(t), \tag{6.1}$$

where $\alpha \in [0, 1]$ is a weighting factor (heuristically set to 0.9) and $FGP(t)$ is the percentage of foreground defined as follows:

Algorithm 6.1 Multi-resolution foreground segmentation.

Input: Current Frame I, desired Level of detail (LoD).
Output: FG mask

begin
  Create the Gaussian pyramidal representation of the current image
  Calculate the foreground/background model in the lowest resolution (n_{max})
  Iterate to the desired output quality (level of the pyramid) as follows:
    Set $n = n_{max} - 1$
    While $n > n_{desired}$
      Up-sample the FG mask of the previous level ($n - 1$)
        Analyze FG regions to calculate Bounding Boxes (ROIs)
        Compute the background segmentation process in the detected ROIs
          Using the foreground/background model of the level
          Using the corresponding image initially built
      Set $n = n - 1$
    Get the desired FG mask
end

$$FGP(t) = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} P_{ij}(t)}{M \cdot N}, \tag{6.2}$$

where M and N are, respectively, the dimensions of the frame and $P_{i,j}(t) \in [0, 1]$ is the binary foreground value of pixel (i, j) at time t.

6.4.2 Feedback-based shadow removal

For the shadow removal task, the *variable-configuration* approach has been implemented to maximize the *agreement between independent detectors* (see subsection 5.4.1). We propose to maximize the similarity between the shadow maps obtained from three independent detectors.

Firstly, the well-known HSV shadow detection algorithm [Prati et al., 2003] is decomposed into the intensity and chrominance parts similarly to [O'Conaire et al., 2007b] (conforming two detectors). The intensity algorithm is controlled by two parameters α and β. In the chrominance algorithm, we make the assumption that shadows also cause a decrease in the pixel's color saturation [Prati et al., 2003] and we only use the *Saturation* as the feature to analyze (discarding the *Hue* data). This assumption is true when the background of the scene presents strong color content. The chrominance algorithm is controlled by the parameter τ_S. The texture algorithm is based on the hypothesis that the change of light inside a shadow is quite smooth [Prati et al., 2003]. In other words, inside a shadow two adjacent pixels would have the same intensity reduction ratio. If there are multiple shadows of the same object, at the border of the intersection

of these shadows, two adjacent pixels may receive two different amounts of light that make the assumption about the consistency of intensity reduction incorrect. This algorithm is controlled by the value d that indicates the maximum percentage of ratio variation among the shadow regions (or surfaces). This agreement process is described as follows:

$$Agreement = F\big(SM_I(\alpha,\, \beta),\, SM_S(\tau_S),\, SM_T(d)\big), \tag{6.3}$$

where the F function computes the agreement of the intensity SM_I, saturation SM_S and texture SM_T shadow maps. For the agreement computation, we propose to use a simple and well known measure of similarity, the correlation between two signals [Chen and Popovich, 2002]. This operation is very efficient in computational terms and it can be easily extended to more than two binary signals (three binary masks in our case) by computing it in pairs.

Then, we propose to maximize this agreement measure by using a gradient ascent algorithm to find the optimum parameters of the three detectors. If the initial values of the parameters are set correctly, we expect that there will be a strong agreement between the three independent detectors. The maximization process is defined as follows:

$$S_i = S_{i-1} + \eta \nabla F(S_{i-1}), \tag{6.4}$$

where $S_i = \{\alpha_i,\, \beta_i,\, \tau_{i,S},\, d_i\}$ is the parameter set that controls the overall shadow detection process at iteration i, $\nabla F(S_i)$ is the gradient of the F function particularized at S_i and $\eta > 0$ is a constant. As the analytical expression of $F(S_i)$ is not available, we obtain $\nabla F(S_i)$ by evaluating $F(S)$ in a neighborhood of S_i. Then, the optimization process is divided in two stages to speed it up. The first stage uses a coarse step η_1 to choose values close to the optimum (until there is no maximization) and the second stage uses a fine step η_2 to choose the optimum values. Fig. 6.5 displays an example of the result of this optimization process.

6.4.2.1 Levels of Detail (LoDs)

The LoDs correspond to different agreement percentages between the detectors. Thus, the LoDs can be viewed as the stopping conditions of the maximization process. Higher LoDs will correspond to higher agreement levels that produce more accurate shadow maps and usually take more time to be accomplished (due to the optimum parameter search).

6.4.2.2 Complexity estimation

As complexity estimator, we use the percentage of blobs correctly classified by the people detector. It is defined as follows:

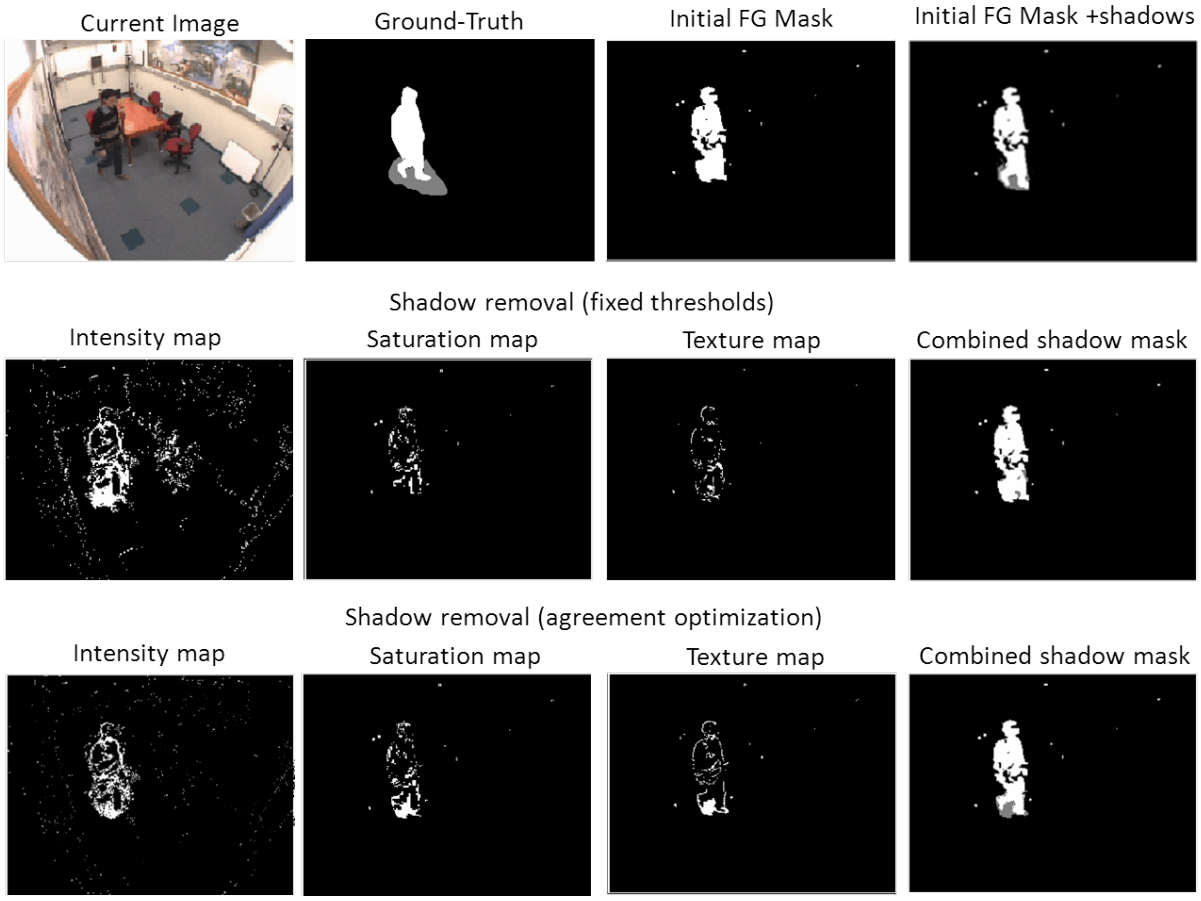


Fig. 6.5. Example of the proposed approach to maximize the agreement of three shadow removal algorithms. The fourth row shows the shadow annotation in the real foreground mask and the shadow masks with fixed and optimum thresholds (gray color indicates shadow detection).

$$PeopleCorrect_i \quad = 1/N \cdot \quad \sum_{j=1}^{N} F\left(PeopleScore_j\right), \tag{6.5}$$

where N is the number of blobs detected in the ith frame, $PeopleScore_j$ is the people likelihood of jth blob and $F(x)$ is a function that assigns a high confidence to blobs correctly classified (understanding well classified as a likelihood close to 0 or 1). It is defined as follows:

$$F(x) = \begin{cases} x & if\ x \geq 0.5 \\ 1-x & if\ x < 0.5, \end{cases} \tag{6.6}$$

where x is the people likelihood of a particular blob.

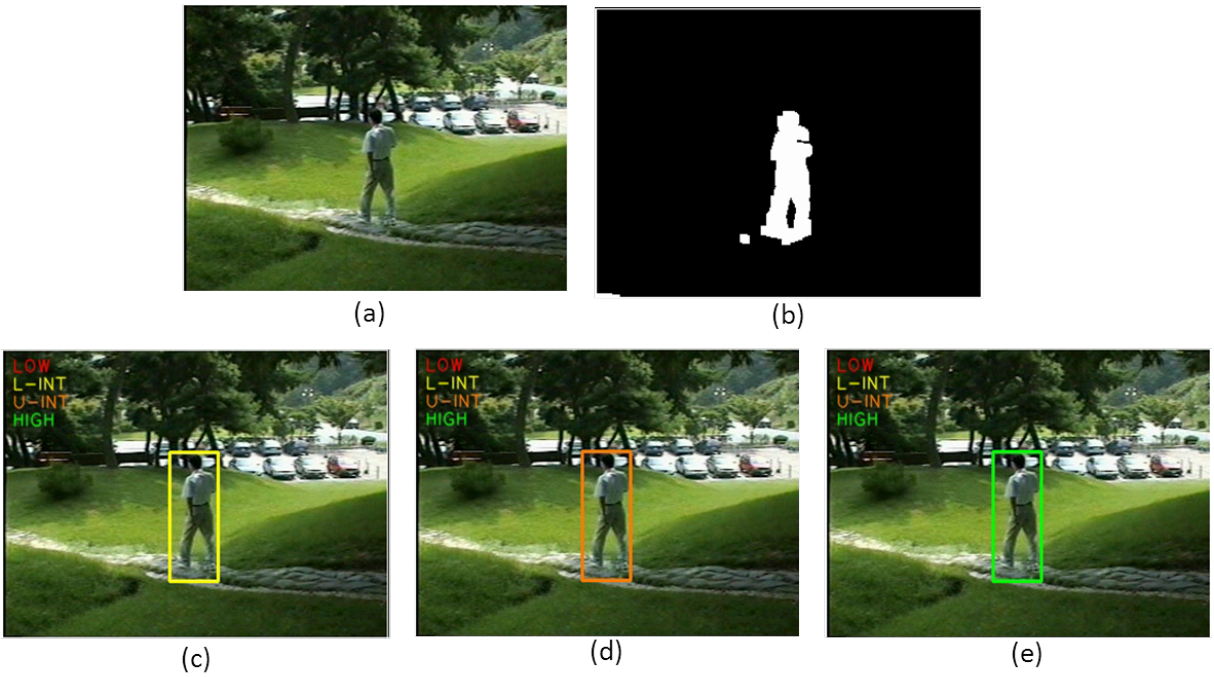


Fig. 6.6. Example for the available LoDs for the people recognition task. Data correspond to (a) the current frame, (b) its foreground mask, (c) the LoD-1 ratio algorithm, (d) the LoD-2 ellipse algorithm and (e) the LoD-3 ghost algorithm.

6.4.3 Feedback-based blob classification

In this stage, we use the i*ncremental focus of attention* approach (see subsection 5.4.1) for people recognition in the detected foreground blobs. We have selected the algorithms based on aspect ratio, ellipse fitting and shoulder location reported in [Fernandez-Carbajales et al., 2008].

6.4.3.1 Levels of Detail (LoDs)

As the output variability only depends on the algorithm being executed, the different LoDs are related with the available algorithms in this analysis component. The algorithms are sorted in increasing computational complexity and executed depending on the requested LoD (that impacts on the quality of the output). As a first approach, we have mapped the following algorithms to each LoD. The *ratio* algorithm is executed for level 1, the *ellipse* algorithm is executed for level 2 and the *ghost* algorithm is executed for level 3. Fig. 6.6 shows an example of these LoDs. As it can be observed, each LoD provides a different result. Furthermore, the number of LoDs can be increased by combining the results of the people detection algorithms, for example, as proposed in [Fernandez-Carbajales et al., 2008].

6.4.3.2 Complexity estimation

The proposed complexity estimator of this stage regards the performance of the current LoD. As the output of this stage are the probabilities of being people for the detected blobs, we use these probability values as indicators of the success of the blob classification stage. We interpret the correct classification when the blob is detected as people or non-people with, respectively, a higher or lower probability. Intermediate probability values are assumed to give a result with high uncertainty. This estimation is described in Eq. 6.5 and it is calculated for all the blobs analyzed with the current LoD and it allows to increase or decrease the LoD of this stage.

6.4.4 Feedback-based event detection

We have introduced feedback in the detection of the modeled events (*Leaves-object, Gets-object, Abandoned-object and Stolen-object*) that requires the use of specific analysis tools (see subsection 6.2.2). In this subsection, we describe two applications for *foreground/background object classification* and *owner search*.

6.4.4.1 Foreground/Background object classification

For this analysis, a similar approach as the blob classification stage has been used (i.e., *incremental focus of attention*). The likelihood of being a foreground or background object is obtained by analyzing the contour and the shape of the detected blob in the current and background images [SanMiguel and Martinez, 2008b]. We propose to have three LoDs as follows: apply contour analysis for the level 1, color analysis for the level 2 and a fusion scheme for level 3 (similarly to [SanMiguel and Martinez, 2008b]). The complexity estimator is also computed as proposed in the blob classification stage. Thus, intermediate probability values provide high uncertainty and require to increase the LoD of this classification problem. This process is iteratively performed until the maximum value is reached (in this case 3).

6.4.4.2 Owner search

For this analysis, the owner of the object of interest is searched in a frame interval previous to the event. This search tries to find blobs with high people likelihood close to the object of interest. A more complex algorithm for owner search tries to search the best owner of the object with a more intensive search. It finds the best owner (closest blob with high people likelihood) and re-evaluates its people likelihood with the highest LoD of the classification stage. Thus, we propose to have two different LoDs (for the basic search and the intensive search). The performance measure that controls the LoD is the people likelihood of the owner.

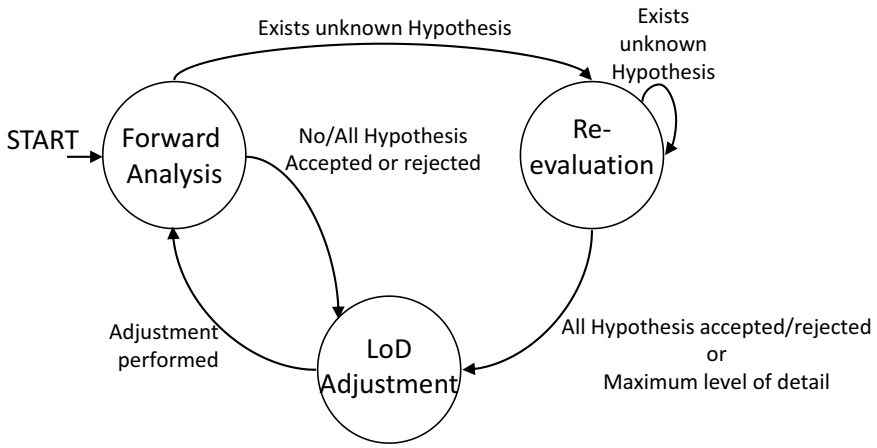


Fig. 6.7. Finite-State-Machine that models the analysis states of the system.

6.5 Feedback information management

After implementing feedback in the processing stages of the video analysis system, a procedure for coordinating all this information needs to be introduced. In this section, we describe this coordination to improve the system performance in terms of accuracy and computational cost.

6.5.1 System manager

A system manager is included into the system to use the information generated by the feedback strategies applied in the system. It is in charge of estimating data complexity, selecting the appropriate LoD for each processing stage and deciding the analysis strategy of the system. In the feedback model of chapter 5, this manager can be seen as the integration of the *estimator* and the *actuator* model components of each processing stage. To describe the current analysis strategy of the system, three states are defined: *Forward Analysis*, *Re-evaluation* and *LoD Adjustment*. We use a Finite-State-Machine for modeling the transitions among stages (shown in Fig. 6.7).

6.5.2 Execution states of the feedback-based system

6.5.2.1 Forward Analysis state

For each frame, the system starts in the *Forward Analysis* state to process the incoming data in sequential mode with the LoDs used in the previous data processing. Then, a status is assigned for the detected events depending on their likelihood. Two confidence values are defined to decide the event detection status of the detection of an event: the event is *accepted* if its value is above 0.7, *rejected* if its value is below 0.3 and declared *unknown* in other case.

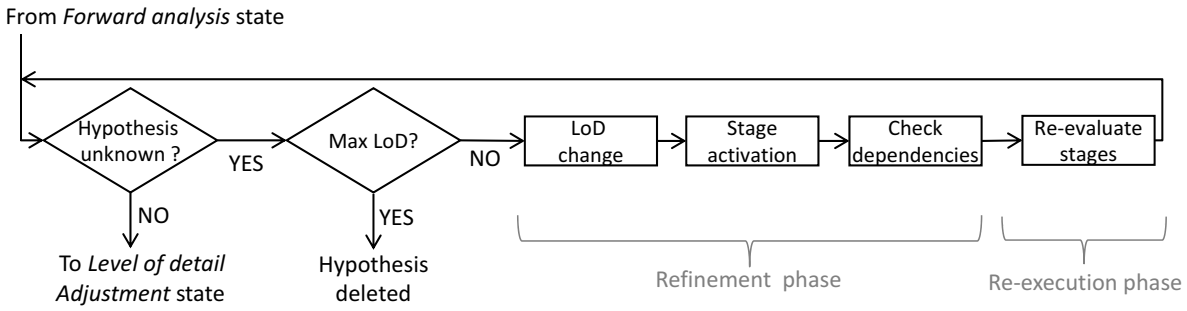


Fig. 6.8. Sequence of operations performed at the *Re-evaluation* analysis state.

6.5.2.2 Re-evaluation state

For every *unknown* event, the system goes to the *Re-evaluation* state for its inspection in order to reach a stable solution (event *accepted* or *rejected*). Firstly, the complexity of the analyzed data is estimated for each processing stage as described in section 6.4. Secondly, the stages with high-complexity (or low performance) are activated and their LoDs are increased to allow a more exhaustive examination of the *unknown* event. Thirdly, the functional dependencies between the activated and the non-activated processing stages are checked to update the necessary intermediate results required for the event detection stage. For example, if the segmentation stage is re-executed, we also might re-calculate the blob analysis stage to detect or eliminate blobs using the new segmentation data. Conversely, if the people recognition stage is re-executed, we only update the people-related event detection results. This sequence of operations is called *Refinement phase*. Finally, the activated processing stages are re-executed (*Re-execution phase*). These two phases are iteratively applied until the event remains in the *unknown* state or there is no possible LoD increase. In the later case, the *unknown* event is removed from the detected event list. After this re-evaluation, the LoDs of each stage return to the initial values in this state for future analysis. Fig. 6.8 summarizes the sequence of performed operations.

6.5.2.3 Level of Detail Adjustment state

Finally, the system goes to the *LoD Adjustment* state when there are no events labeled as *unknown* or no detected events. In this state, the objective is to efficiently use the available resources (e.g, computational effort) by adapting the LoD of the processing stages to the complexity of the data maintaining an acceptable performance level (that is, the event detection accuracy) under different complexities of the sequence (e.g., crowded situations, object occlusions). For each processing stage, we model its LoDs as the states of a FSM. Their transitions are based on applying thresholds (empirically determined) on conditions over the complexity estimation over the input and output data. Fig. 6.9 depicts the FSMs for each processing stage.

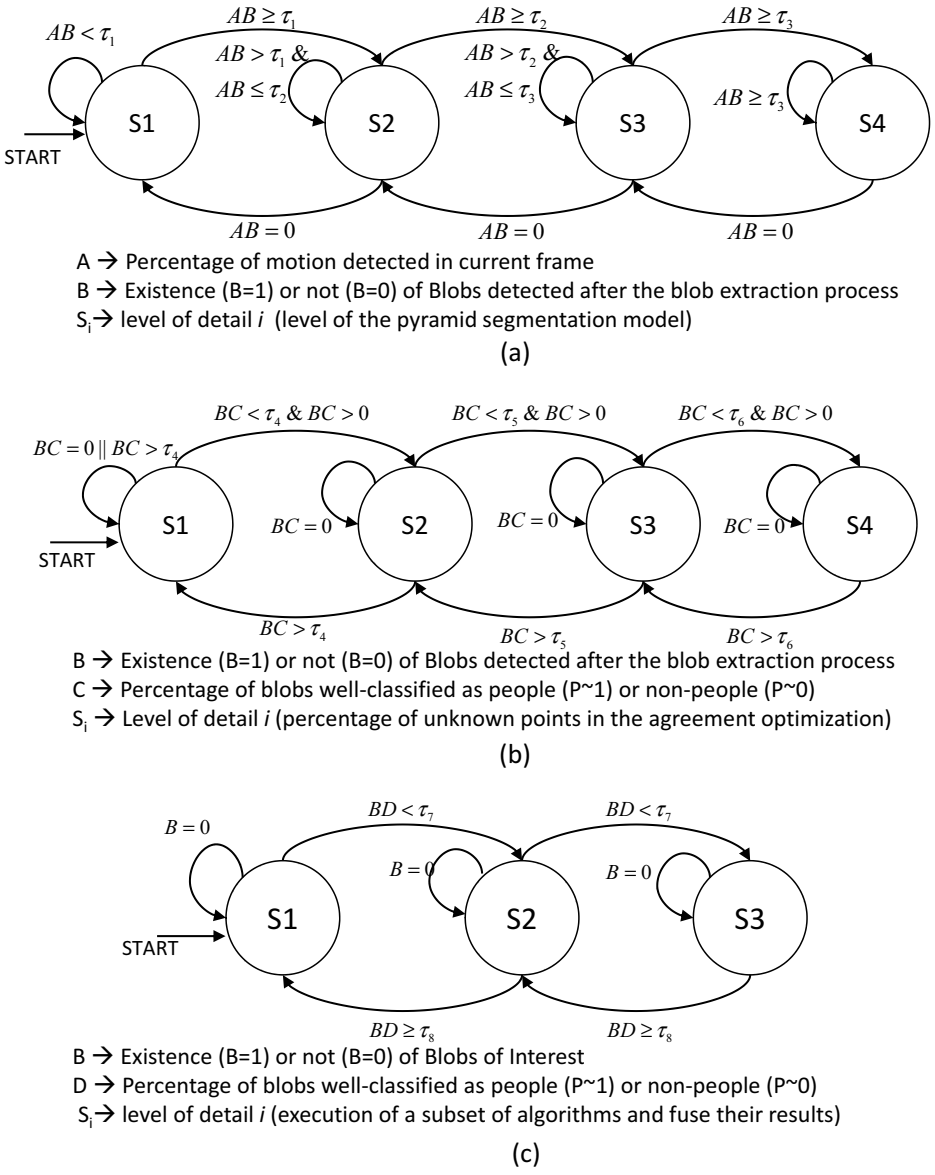


Fig. 6.9. Finite-state-machines to define LoD adjustment for (a) the foreground detection, (b) the shadow removal and (c) the blob classification processing stages. The event detection FSM has the same structure as the blob classification one differing in the meaning of condition D and the interpretation of the used thresholds.

The *foreground detection* FSM (Fig. 6.9(a)) has four states defined by the levels of the pyramidal segmentation model. A set of thresholds (τ_1, τ_2 and τ_3) is defined to model the state transitions based on the percentage of foreground motion as defined in 6.4.1.2. These thresholds capture when a motion percentage is relevant for the video sequence. The *shadow removal* FSM (Fig. 6.9(b)) has four states corresponding to the different levels of agreement between the

applied detectors. A set of thresholds (τ_4, τ_5 and τ_6) is defined to model the state transitions capturing the relevant percentage of blobs correctly classified. The *blob classification* FSM (Fig. 6.9(c)) has three states defined by the application of the fusion of three detectors. The associated transitions are based on two thresholds (τ_7 and τ_8) that define the relevant percentage of blobs correctly classified as people or non-people. The event detection FSM has three states that define the application of a single detector (using gradient, color or contour information). The associated transitions are based on the accuracy of the detector output using two thresholds (τ_9 and τ_{10}). Additionally, all FSMs try to decrease the level of detail if there are no detected blobs in the current frame.

6.6 Experimental results

We present a comparative evaluation[2] of the proposed feedback-based and the base event detection systems. Both systems have been implemented in C++, using the OpenCV library[3]. Tests were executed on a Pentium IV with a CPU frequency of 2.8 GHz and 1GB RAM. To our best knowledge, most of the feedback-based approaches are focused on improving single processing stages instead of complete systems and, therefore, a state-of-art comparison is not possible.

6.6.1 Dataset

Experiments were carried out on several sequences from AVSS2007[4], PETS2006[5] and PETS2007[6], VISOR[7], CANDELA[8], WCAM[9] and CVSG[10] datasets. The four selected events (*Leaves-object*, *Gets-object*, *Abandoned-object* and *Stolen-object*) have been manually annotated using the Viper toolkit [Doermann and Mihalcik, 2000].

For performance evaluation, we have divided the test sequences into different complexity categories depending on the difficulty of the foreground object extraction and the background complexity as they affect most of the algorithms applied in the processing stages. The former regards the difficulty to extract moving objects in a scene. It is related with the number of objects, occlusions, lighting changes and people detection difficulty. The latter considers the presence of edges, multiple textures and objects belonging to the background (e.g., waving trees, water surface). Table 6.1 lists a content description and Fig. 6.10 shows some sample frames.

[2] Additional results and test data can be found at http://www-vpu.eps.uam.es/publications/FeedbackEventDetection

[3] http://sourceforge.net/projects/opencvlibrary/

[4] http://www.avss2007.org/

[5] http://www.cvg.rdg.ac.uk/PETS2006/

[6] http://www.cvg.rdg.ac.uk/PETS2007/

[7] http://www.openvisor.org/

[8] http://www.multitel.be/~va/candela/abandon.html

[9] http://wcam.epfl.ch/

[10] http://www-vpu.eps.uam.es/CVSG/

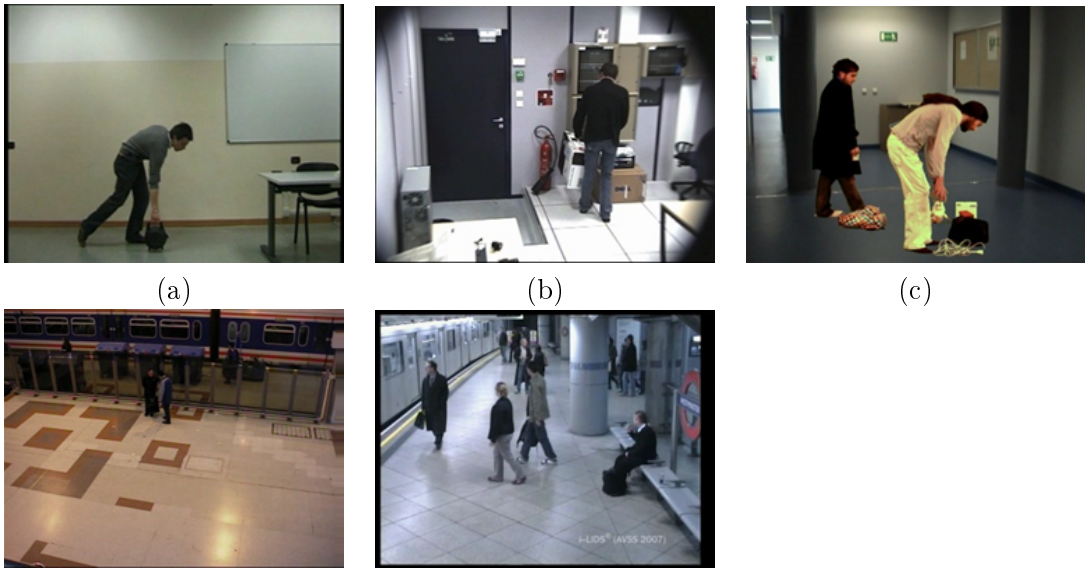


Fig. 6.10. Sample frames of the collected dataset for the evaluation of the proposed feedback event detection for category (a) C1, (b) C2, (c) C3, (d) C4 and (e) C5.

Category	Length	Complexity	
		FG	BG
C1	21500	Low	Medium
C2	25650	Low	High
C3	110255	Medium	Medium
C4	162325	Medium	High
C5	95630	High	Medium

Table 6.1: Test Sequence Categorization

The parameters of the transitions of the FSMs defined in subsection 6.5 were empirically obtained considering the selected experimental data. Two sets of thresholds were defined for the FSM of foreground segmentation considering the two available views of the test sequences: close-view ($\tau_1 = 0.02$, $\tau_2 = 0.1$ and $\tau_3 = 0.3$) and far-view ($\tau_2 = 0.01$, and $\tau_3 = 0.03$). For the other three FSMs, a unique set of thresholds was defined as their analysis does not directly depend on the sequence view ($\tau_4 = 0.75$, $\tau_5 = 0.85$, $\tau_6 = 0.9$, $\tau_7 = 0.5$, $\tau_8 = 0.75$, $\tau_9 = 0.5$ and $\tau_{10} = 0.75$). This set captures when a shadow or blob is difficult to analyze and, therefore, an analysis with higher level of detail is needed. In case of the analysis of a different scenario, a new set of thresholds has to be obtained considering the data complexity and the analysis capabilities of the levels of detail for each processing stage.

Category	Base System						Feedback System					
	TP	FN	FP	R	P	F_{score}	TP	FN	FP	R	P	F_{score}
C1	20	0	6	1	.77	.87	20	0	2	1	.91	.95
C2	39	2	15	.95	.72	.81	38	3	10	.92	.79	.85
C3	480	169	287	.73	.62	.67	468	181	198	.72	.70	.71
C4	258	201	536	.56	.28	.37	230	229	303	.50	.43	.46
C5	60	109	168	.35	.26	.30	55	114	125	.32	.30	.31
Total	857	481	1012	.64	.45	.53	811	527	638	.60	.55	.57

Table 6.2: Accuracy of the base and the proposed feedback-based systems in terms of correct detections (TP), miss-detections (FN), wrong detections (FP), Precision (P) and Recall (R).

6.6.2 Performance evaluation criteria

As evaluation criteria, we have used the same one described in subsection 4.6.1. Thus, an annotated event is detected if there is a detection of the same type that satisfies the following constraints: its likelihood is higher than 0.7, the overlapped duration in frames between them is more than 50% and the mean overlapped area between them is more than 50% (calculated in the overlapped frames). Moreover, event detection accuracy is evaluated with Precision (P) and Recall (R). We also use the F-score measure to combine Precision and Recall. It is defined as follows:

$$F_{score} = 2 \cdot \frac{P \cdot R}{P + R}. \tag{6.7}$$

6.6.3 Results and discussion

Table 6.2 summarizes the obtained detection results for the base and the feedback systems. As it can be observed, the proposed feedback strategies improve the precision with a small penalization in the recall measure for each category. Firstly, the re-evaluation analysis state improves the system precision because of the re-inspection of *unknown* events (initially discarded by the base system) with a higher LoD (only when it is required). Secondly, the recall measure is slightly reduced with the feedback strategies because of the use of the lowest levels of detail. This might affect the accuracy of low-level processing stages by missing some data of interest (e.g., moving blobs in foreground segmentation). Additionally, it can be seen that our approach achieves better results in sequences with less foreground object extraction complexity and that it is robust to background complexity because of the algorithms that use background information are robust to highly textured and blurred backgrounds (obtained with the multi-resolution approach applied in the foreground detection stage). Globally, the F_{score} improvement was around the 8%.

Table 6.3 illustrates a computational cost comparative. It demonstrates that the feedback-

Processing Stage	Average proc. time (ms) 720x576		Average proc. time (ms) 360x288	
	Base	Feedback	Base	Feedback
Foreground segmentation	61.93 (34%)	22.43 (20%)	15.45 (41%)	5.21 (20%)
Shadow Detection	65.50 (35%)	29.67 (25%)	16.42 (44%)	12.44 (48%)
Blob Extraction	1.31 (1%)	1.35 (1%)	0.36 (1%)	0.20 (1%)
Tracking	2.90 (2%)	3.21 (3%)	0.41 (1%)	0.43 (2%)
People Detection	13.15 (7%)	14.85 (13%)	1.37 (3.5%)	1.82 (7%)
Feature Extraction	12.74 (7%)	10.64 (9%)	2.81 (7.5%)	2.67 (10%)
Event Detection	27.13 (15%)	29.96 (26%)	0.85 (2%)	2.14 (8%)
Total	184.69 (100%)	117.22 (100%)	37.69 (100%)	25.68 (100%)
Average fps	5.41	8.53	26.52	38.94

Table 6.3: Computational cost comparison between the base and the feedback systems. Results are reported in terms of the two resolutions (360x288 and 720x576) of the selected test sequences.

based system reduces the overall computational cost without reducing the system detection capabilities (see Table 6.2). Pixel-based stages (background subtraction and shadow detection) present a high computational cost reduction because they are applied to full frame resolution in the base system (resulting in high computational costs) and the feedback strategies allow to focus their effort in the regions of interest determined by the lowest LoDs of the foreground detection stage. Object-based approaches (people detection and event detection) present a variable computational cost depending on the sequence being analyzed. Globally, the impact in the computational cost reduction is higher for the pixel-based approaches than for the object-based approaches. Furthermore, this reduction relies on the assumption that video-surveillance sequences present long periods of low activity and low LoDs can be used.

On the other hand, high-motion or high-activity periods do not necessarily imply more refinement (an increase in the LoD and, therefore, more computational resources) as events could be correctly detected with low LoDs. Nevertheless, high-activity usually means more events of interest in practical situations. In this case, it is very likely that some of the detected events will be *unknown* requiring a LoD increase. However, the maximum computational effort of the feedback system (when all processing stages operate at their maximum LoD) is similar to the base system (actually it is slightly higher because of the system manager). In conclusion, the computational cost reduction may benefit the overall performance in systems that monitor multiple cameras (increasing the analysis effort in the cameras detecting high movement and reducing the computational cost for those detecting low movement).

Fig. 6.11 depicts an example of the event detection improvement by using feedback strategies for the sequence *S5_ T1_ G3* from the PETS2006 dataset (accepting the hypothesis in this case). An initial likelihood of the *Abandoned-object* hypothesis (event) is computed and it is marked as

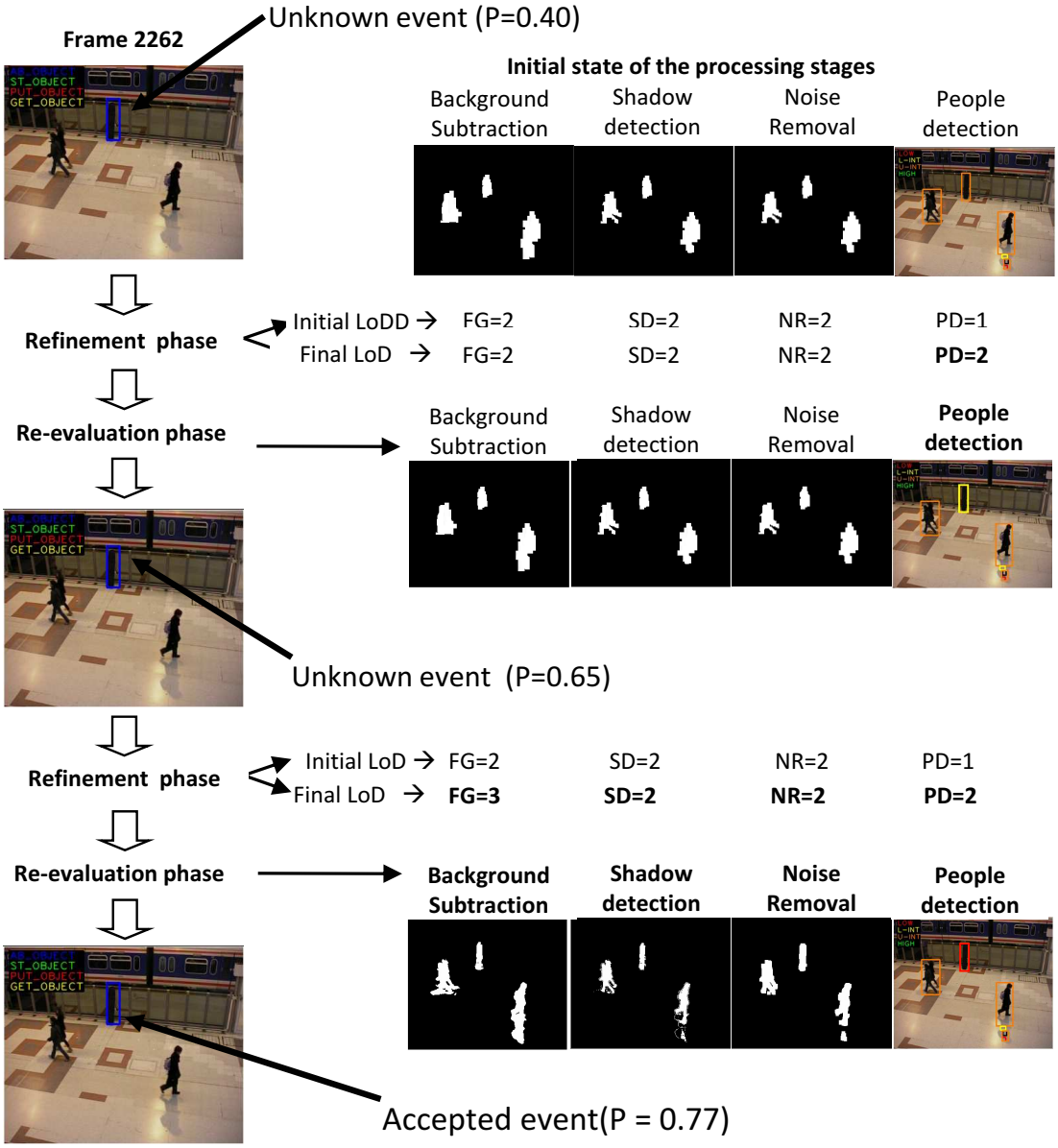


Fig. 6.11. Example of the event detection improvement with the proposed feedback-based re-evaluation strategy for the frame 1664 of the test sequence *S5-T1-G3* from PETS 2006 dataset (in the *Refinement phase*, the LoD changes for each processing stage are marked in bold)

unknown as its value is between 0.3 and 0.7. Then, the system manager starts the *Re-evaluation* state to examine the *unknown* event with more detail. As a first attempt, the level of detail of the people detection stage is increased in the *Refinement phase* (see Fig. 6.8) and the likelihood of the event is again computed. As it remains in the *unknown* state, the system manager decides to increase the level of detail of the foreground detection and the people detection stages in the

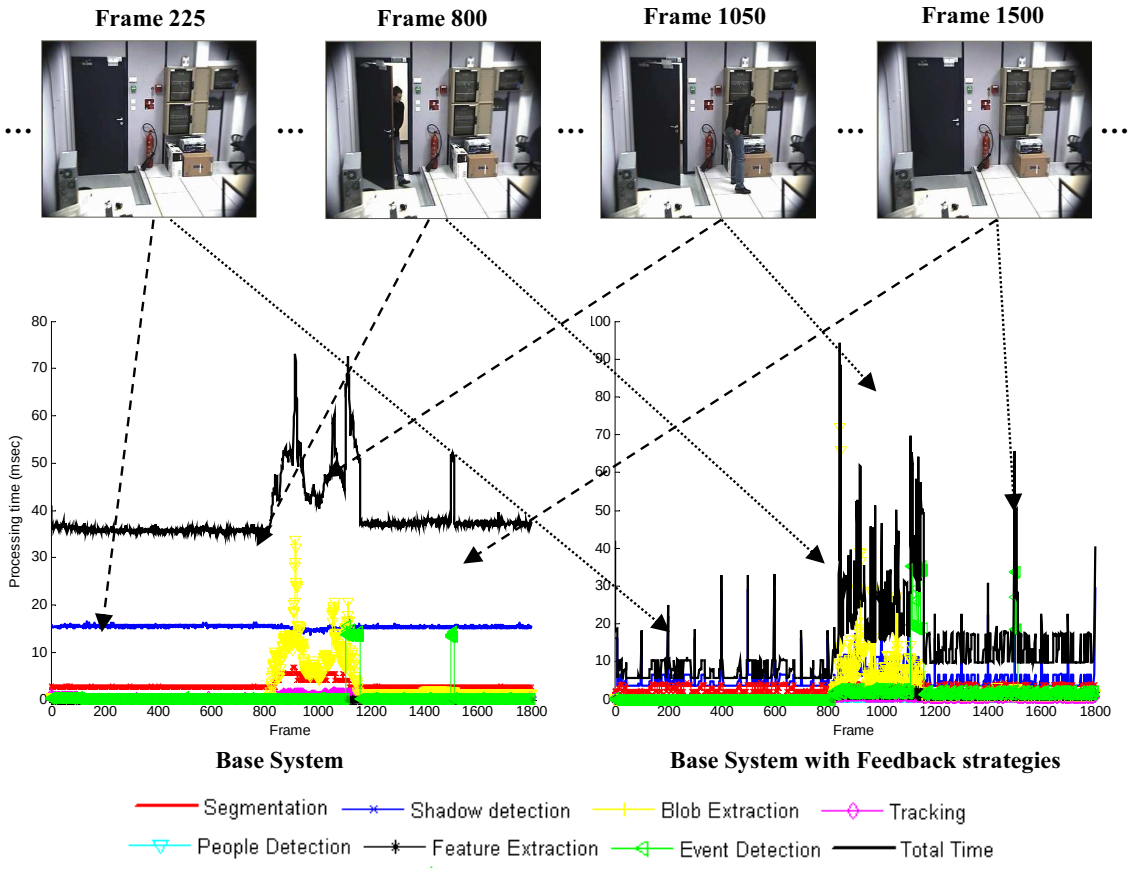


Fig. 6.12. Computational cost example for the base and the feedback systems for the test sequence *Indoor_ activity_ stolen_ object_ cif* from the WCAM dataset

Refinement phase. Additionally, the dependencies between the processing stages are checked (see Fig. 6.8) determining that the shadow removal stage has to be re-evaluated due to their dependence with the foreground detection stage. Finally, the likelihood of the event is computed and it is *accepted* because its likelihood is above 0.7.

Fig. 6.12 illustrates an example of the computational cost reduction by using feedback strategies for the sequence *Indoor_ activity_ stolen_ object_ cif* of the WCAM dataset. It can be observed that the system manager reduces to the minimum requirements the computational complexity when there are not objects of interest in the scene (frame 225), gradually increases the computational requirements of the analysis stages when video objects appear in the scene (a person enters the room around frame 800), analyzes the objects of interest that may produce events to detect with high LoD (the person steals an object around frame 1000) and, finally, reduces the computational requirements to intermediate levels of detail in order to monitor the stationary object of interest in the scene (the blob produced by the stolen object).

6.7 Summary and conclusions

In this chapter, we have presented a feedback-based approach to detect events in video surveillance supported by the core feedback scheme defined in chapter 5 that is based on variable level of detail and the measurement of complexity of the data analyzed. Three feedback-based strategies are proposed and implemented in the processing stages of a typical video surveillance system to improve the accuracy of the event recognition stage and to adjust its computational cost in simple and complex situations.

Experimental results showed that the proposed approach maintained the initial event detection results increasing the precision (reducing the percentage of false events) whereas slightly reducing the recall (missing few events). These effects on the precision and recall were due to the use of the *Re-evaluation* state (for analyzing *unknown* events) and the *Level of Detail Adjustment* state (for adjusting the computational cost). Moreover, a high computational cost reduction was achieved due to the inclusion of the *Level of Detail Adjustment* state being one of the main advantages of the proposed approach. Feedback strategies applied to pixel-based processing stages were useful to dramatically reduce their associated computational cost. On the other hand, feedback strategies applied to object-based processing stages improved the accuracy of the results obtained. As a conclusion, the achieved results could be very useful to add more processing capabilities (e.g., algorithms) in video processing systems whilst maintaining the same or lower computational cost. For example, multi-camera setups may benefit of increasing the number of cameras analyzed per processing unit by using the proposed feedback-based approach.

As observed in the experiments, the current feedback model exhibited some limitations. Firstly, it is not able to generate new hypothesis (e.g., detect foreground blobs that are initially missed) being limited to the refinement of initial hypothesis (e.g., blobs, events). In other words, it only allows to improve Precision. Secondly, the iterative nature of the feedback model may restrict the real-time operation if several LoDs are available for analysis. In this situation, the analysis of complex data may require to perform several iterations in the feedback loop for finding the optimum LoD. Hence, there is a tradeoff between the computational cost and the number of implemented LoD. The last limitation consists in the assumption of accurate measures for evaluating input complexity and output quality. Both task are very difficult requiring to be properly studied. Specifically, the evaluation of foreground segmentation and tracking was observed as a critical issue to solve as most of the video analysis systems rely on both stages.

In the chapters 7 and 8, we address the estimation of the output quality in real conditions (i.e., without ground-truth information) in order to develop robust feedback-based systems in real-world applications. The aim of this study is to provide algorithms able to operate similarly to the approaches for evaluation with ground-truth data. In particular, we focus on the foreground segmentation and object tracking stages.

Chapter 7

On quality estimation without ground-truth for foreground segmentation and tracking

7.1 Introduction[1]

Foreground segmentation and tracking are the basic stages for many video applications. Several approaches have been proposed based on the widely used *Background Subtraction* technique [Benezeth et al., 2010] and the *Particle Filter framework* [Maggio and Cavallaro, 2011] for, respectively, foreground segmentation and object tracking. Moreover, these approaches usually operate in different conditions generated by indoor, outdoor or crowded environments. Hence, no single algorithm can perform perfectly in all situations and failures are expected in real scenarios.

For developing robust approaches (e.g., by using the feedback model of chapter 5), an estimation of the output quality is required to evaluate the algorithm performance. In this chapter we propose a hierarchical organization of the related literature for evaluating the output quality of video object segmentation and tracking approaches without ground-truth information. Furthermore, we present a comparison of the most representative approaches on public available datasets. The result of this evaluation is a recommendation on which output quality measures perform better under specific characteristics of the sequences that affect the algorithm accuracy. These measures can be used in the feedback model proposed in chapter 5.

The remainder of this chapter is organized as follows: Section 7.2 describes the related work.

[1] This chapter is based on the publications "*J.C. SanMiguel and J.M. Martínez. On the evaluation of background subtraction algorithms without ground- truth. In Proc. of the IEEE Int. Conf. on Advanced Video and Signal based Surveillance, pp. 180-187, Boston (USA), 1-3 Sept. 2010*" and "*J.C. SanMiguel, A. Cavallaro, and J.M. Martínez. Evaluation of on-line quality estimators for video object tracking. In Proc of the IEEE Int. Conf. on Image Processing, pp. 825-828, Hong Kong (China), 26-29 Sept. 2010*".

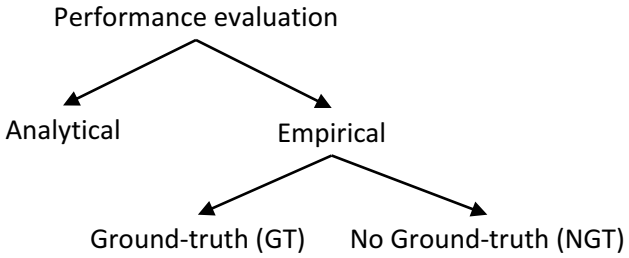


Fig. 7.1. Methodologies for performance evaluation of video object segmentation and tracking.

The selected approaches for the evaluation of the output quality of foreground segmentation and tracking are presented in, respectively, sections 7.3 and 7.4. Section 7.5 discusses the evaluation methodology. Section 7.6 describes the optimum parameter study. Section 7.7 shows the experimental results and, finally, section 7.8 describes the conclusions and future work.

7.2 Related work

The evaluation of the output quality of video object segmentation and tracking algorithms is crucial to estimate their accuracy and to tune their parameters for optimal performance. Although analytical approaches exist, this evaluation is typically performed by comparing the obtained results with manual annotations (or ground-truth, GT). However, manual annotation is time-consuming and prone to human error. It usually covers a small set of video sequences only representing a small percentage of data variability. This limitation complicates the extrapolation of the performance evaluation results to new (unlabeled) sequences. Moreover, evaluation using ground truth is not feasible for on-line performance analysis. Conversely, the evaluation not-based on ground-truth (NGT) is a desirable option to overcome these limitations. A hierarchical organization of these evaluation methodologies is given in Fig. 7.1. In this section, we review the related NGT literature for evaluating foreground segmentation and object tracking.

7.2.1 Foreground segmentation quality estimation

The evaluation of foreground segmentation has been mainly approached by empirical methods, although there are some analytical proposals like [Gao et al., 2000]. Moreover, empirical evaluation can be divided into the use (or not) of annotations of foreground objects. GT evaluation, also known as *relative evaluation* [Correia and Pereira, 2003] or *empirical discrepancy evaluation* [Zhang et al., 2008], is based on directly measuring the deviation between the segmentation results (e.g., binary masks) and the manual annotations of foreground objects. For instance, [Herrero and Bescos, 2009] analyzed the performance of seven background subtraction methods by counting the number of pixels correctly and erroneously classified as foreground and back-

ground under different test conditions. Moreover, [Nascimento and Marques, 2006] defined more sophisticated statistics using GT information with the aim to detect the split and merge of foreground objects, false alarms and the detection of failures. Additionally, the visual relevance can be considered to weight the segmentation errors [Villegas and Marichal, 2004]. NGT evaluation, also known as *stand-alone evaluation* [Correia and Pereira, 2003] or *empirical goodness evaluation* [Zhang et al., 2008], is based on inspecting desired properties of the empirical results. These methods assume prior information about expected segmentation results (e.g., matching of color and object boundaries). Among the existing NGT approaches, we differentiate region-based, model-based and assisted-based evaluation methods.

7.2.1.1 Region-based evaluation

Region-based measures inspect the properties of the internal and external regions defined by the boundary of the segmented foreground region.

Internal measures (IM) study the homogeneity of object features within the object boundary. For instance, [Weszka and Rosenfeld, 1978] used the difference (in gray-level) between the original and the segmented images as a measure of segmentation quality. In addition, [Correia and Pereira, 2003] proposed several spatial features like circularity, elongation and compactness of the objects as object homogeneity features. However, these features rely on the temporal correspondence of the video object (tracking). The same authors also defined motion uniformity as temporal homogeneity feature not restricted to use tracking data. Moreover, [Snidaro et al., 2007b] described blob-based measures based on their gray-level difference and connectivity. It should be noted that unsupervised evaluation of still image segmentation has received more attention than the video object segmentation during last years [Zhang et al., 2008]. However, its applicability is limited to regions with uniform properties. As video objects are usually composed of various color regions (e.g., people), it is expected that these unsupervised approaches will fail.

On the other hand, *external or contrast* measures (EM) consider feature differences between the internal and external regions defined by object boundaries. For instance, color contrast is proposed in [Erdem et al., 2004a; Chabrier et al., 2006]; moreover, motion contrast is also defined in [Erdem et al., 2004a]. Both contrast measures are calculated in the neighborhoods of each boundary pixel. [Piroddi and Vlachos, 2006] described an improved version of the pixel neighborhood used in [Erdem et al., 2004a] to address the problem of unreliable and unavailable feature estimations. [Kubassova et al., 2008] defined the edge profiles to analyze color differences under low contrast conditions. However, these measures are demonstrated in low-populated sequences with high contrast between moving objects and scene background. The extrapolation of the obtained results to more complex sequences is not straightforward.

Category	Sub-category	Features	Metrics
Region	Internal	Circularity, elongation and compactness	Euclidean
	External	Gray-level, color, motion and edges	Euclidean
Model	-	Model similarity (e.g., people, car)	Thresholding
Assisted	-	Size, position & appearance	Correlation

Table 7.1: Foreground segmentation quality estimators not-based on ground-truth.

7.2.1.2 Model-based evaluation

Model-based measures examine the impact of the segmentation results on the following analysis stages (e.g., object classification). They are based on the availability of video object models (e.g., person) or artifacts (e.g., shadows) expected to appear in the video sequence. This evaluation approach is useful to measure if the segmented regions satisfy the system requirements (e.g., people detection). Within this category we can find some proposals. [Harville, 2002] used different high-level modules to detect expected foreground objects (people, non-people and illumination changes) and to estimate segmentation accuracy at bounding box level in order to feedback the segmentation module. Moreover, [Cheung and Kamath, 2005] validated foreground masks by building a simple moving object model using foreground and background statistics as well as the frame difference. A block-based human model is proposed in [Rincon et al., 2007] to assess the accuracy of segmentation masks and to correct segmentation errors.

7.2.1.3 Assisted-based evaluation

Assisted-based measures use complementary algorithms to estimate foreground segmentation quality. The key idea is to automatically build an approximation of the GT data to estimate foreground segmentation quality. The expected accuracy is low because they are very dependent on the results of the complementary algorithm. For example, [Garcia and Bescos, 2008] evaluated a Single Gaussian Background subtraction stage with a Frame Difference technique. Similarly, [O'Conaire et al., 2007a] proposed to analyze visual and infrared data for object segmentation. [Goldmann et al., 2008] constructed a GT estimation by using a region-based segmentation algorithm and matched the boundary of segmented video foreground objects and obtained regions.

Table 7.1 summarizes the NGT categories for estimating foreground segmentation quality.

7.2.2 Video object tracking quality estimation

Common tracking performance evaluations also use GT or *empirical discrepancy methods* [Maggio and Cavallaro, 2011] that compare off-line ground-truth data with the estimated target state. For example, [Kasturi et al., 2008] defined spatio-temporal measures to compare tracking data

with manual annotations. To extend the applicability of performance evaluation, NGT or *empirical standalone methods* (ESM) for track-quality estimation without ground-truth data have been defined for large unlabeled datasets, self-tuning (automatic control via on-line analysis), comparative ranking and tracker fusion. ESMs can be classified into three main categories, namely, trajectory-based, feature-based and hybrid.

7.2.2.1 Trajectory-based evaluation

Trajectory-based measures use information from the estimated trajectories to quantify the quality of a tracker and can in turn be grouped into three sub-categories: model-based, forward, and reverse measures. *Model-based measures* (MM) rely on on-line learning of trajectory models. Track quality is computed as the similarity between models and new object trajectories [Piciarelli et al., 2005; Hall, 2006]. MMs need a considerable amount of data to acquire the models thus limiting their application for evaluation. *Forward-based measures* (FM) threshold features extracted from the estimated trajectory in short time intervals. Examples of features are the trajectory length [Chau et al., 2009] and the smoothness of the target velocity [Wu and Zheng, 2004; Chau et al., 2009; Doulamis, 2010] or of the direction of change [Polat et al., 2001; Black et al., 2003; Wu and Zheng, 2004]. FMs generally provide only a binary decision and they are application-dependent thus limiting their field of applicability. *Reverse measures* (RM) rely on the time-reversibility of the motion of physical objects. A tracking analysis in the reverse time direction is applied to measure track quality with different strategies, such as on a frame-by-frame basis template matching [Liu et al., 2008] or on the full trajectory length using the Kanade-Lucas-Tomasi tracker [Kalal et al., 2010] or a particle filter [Wu et al., 2010]. This idea can be extended by reflecting the two trackers analysis to the specific time instant to be evaluated [Pan et al., 2009b]. Although RMs have been found to be preferable to other trajectory-based approaches, their applicability is limited to short sequences as they suffer from error accumulation (short-length versions) or are non-computationally feasible (full-length versions).

7.2.2.2 Feature-based evaluation

Feature-based measures analyze internal stages or the output of a tracker and quantify either features difference or features consistency. Methods based on *feature differences* (FD) estimate track quality by considering feature variations related to background/foreground color differences [Collins et al., 2005; Han et al., 2008] or boundary contrast along the target contour [Erdem et al., 2004a,b]. However, as this variation cannot be guaranteed in all kind of scenarios (e.g. targets similar to the background), FDs are application dependent and non-adequate for assessing general-purpose trackers. Methods based on *feature consistency* (FC) compute statistics to validate feature values over time and may look at shape [Correia and Pereira, 2002], scale [Wu and Zheng, 2004] or appearance consistency [Black et al., 2003; Erdem et al., 2004a; Motamed,

Category	Sub-category	Features	Metrics	Trackers
Trajectory	Forward	Size & position	Euclidean	D & P
	Model	Position	Euclidean	D & P
	Reverse	Position & state-space model	Mahalanobis & Euclidean	D & P
Feature	Difference	Position & contour	Bhattacharyya & Euclidean	D & P
	Consistency	Size, appearance & state-space model	Inf. Theory & change detection	P
Hybrid	-	Size, position & appearance	Euclidean	D & P

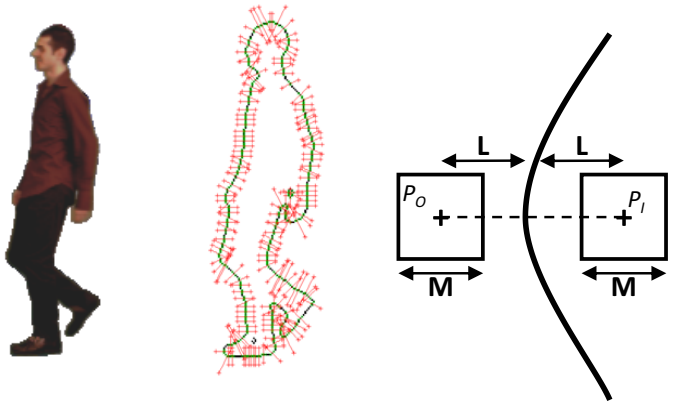
Table 7.2: Track quality estimators not-based on ground-truth. (D: Deterministic; P: probabilistic).

2006; Nickels and Hutchinson, 2002; Loutas et al., 2004]. Furthermore, probabilistic trackers provide an estimation of the target state that is exploited to compute statistics related to the observation likelihood [Lu et al., 2004; Badrinarayanan et al., 2007a; Vaswani, 2007], the covariance of the target state [Badrinarayanan et al., 2007b; Bagdanov et al., 2007; Maggio et al., 2007; Vaswani, 2007] or statistical tests (e.g., Chi-Square [Van der Heijen, 2006], Kolmogorov-Smirnov [Powers and Pao, 2006]). FCs based on probabilistic tracking using the target state representation outperform other approaches. However, they fail when the target moves across areas with varying levels of clutter that affect the observation likelihood. Moreover, when the tracker follows distractors (objects with similar features to that of the target) it might maintain the same level of observation likelihood. A mechanism to determine these tracking conditions on-line is therefore important for an evaluation tool to work adaptively.

7.2.2.3 Hybrid evaluation

Finally, *hybrid* measures combine previously described approaches. Smoothness in both direction and motion can be combined with color consistency [Black et al., 2003]. Likewise, an equally weighed combination of time-reversibility evaluation and feature difference using color histogram and sum of square differences error estimation can be used [Pan et al., 2009a; Kalal et al., 2010]. Finally, multiple measures such as motion smoothness, trajectory complexity, shape and color consistency can be used to produce multiple track quality estimations [Wu and Zheng, 2004; Chau et al., 2009]. Although this combination is recently gaining attention to overcome the existing problems, current hybrid approaches only propose straightforward combinations of trajectory and feature based evaluation methods limiting their success to simple situations.

Table 7.2 summarizes the NGT categories for estimating object tracking quality.



(a) (b) (c)

Fig. 7.2. Boundary-based contrast scheme proposed by [Erdem et al., 2004a]. (a) Segmented object, (b) its boundary with the normal lines and (c) a zoom on a boundary pixel location.

7.3 NGT measures for foreground segmentation quality

We have selected the *region-based* measures based on color and motion contrast proposed by [Erdem et al., 2004a]. We have decided not to use measures from the other two described categories (*model* and *assisted*) because the constraints introduced (model of foreground regions and accuracy of the additional algorithm) are hard to satisfy. On the contrary, the matching of object and color region boundaries is usually satisfied for the video analysis domain. Consequently, the segmentation algorithms selected for the experiments do not include any foreground modeling.

7.3.1 Region-based measures

The first measure selected is the color contrast along the boundary [Erdem et al., 2004a]. It is based on defining normal lines of length $2L+1$ for each boundary pixel and comparing the color differences between the initial (P_I) and ending (P_O) points of each normal line. The neighborhood of these pixels is also considered by using a window of size $M \times M$. The scheme is depicted in Fig. 7.2. It proposes to estimate the segmentation quality of each boundary pixel using the Boundary Spatial Color Contrast feature defined as follows: .

$$BSCC(t;i) = \frac{\left\| C_O^i(t) - C_I^i(t) \right\|}{\sqrt{3 \cdot 255^2}}, \tag{7.1}$$

where $C_O^i(t)$ and $C_I^i(t)$ are the mean colors calculated in the $M \times M$ neighborhood of the points P_I and P_O (using the RGB color space quantified into 256 levels) for each *i-th* boundary pixel of the foreground region at time t. This measure ranges from 0 to 1 depending if all pairs of mean colors belong to, respectively, the same or different color regions.

Then, it proposes to evaluate the foreground segmentation of each object, O_j, and to combine the segmentation of multiple objects as follows:.

$$DC1_{O_j}(t) = \frac{1}{K_t}\sum_{i=1}^{K_t} BSCC(t;\, i,\, j), \tag{7.2}$$

$$DC1(t) = \min_j\big(DC1_{O_j}(t)\big), \tag{7.3}$$

where K_t is the total number of boundary pixels, BSCC is the spatial color contrast of the *i-th* boundary pixel of the *j-th* foreground region being analyzed. Its value ranges from 0 (lowest segmentation quality) to 1 (highest segmentation quality).

Additionally, [Erdem et al., 2004a] used this measure to detect incorrectly segmented boundary pixels if they are above a certain threshold, T_1. Thus, a second measure of segmentation quality could be derived by counting the correctly segmented boundary pixels as follows .

$$DC2_{O_j}(t) = \frac{\#(BSCC(t;\, i,\, j) > T_1)}{K_t}, \tag{7.4}$$

$$DC2(t) = \min_j\big(DC2_{O_j}(t)\big). \tag{7.5}$$

The main advantages of these measures are their low complexity and their possibility to detect failures at finer level (boundary pixel). These aspects make the measure useful for its use to adapt or feedback real-time video segmentation algorithms to improve the segmentation performed. The parameters (of this measure) to study are the normal line length L, the size M of the window around P_I/P_O points and the threshold, T_1, used in the DC2 measure. As it can be observed the two measures based on color, DC1 and DC2, fall in the range $[0, 1]$.

The third measure is based on the motion difference along the object boundary [Erdem et al., 2004a]. Similarly to the color-based measure, normal lines of length $2L + 1$ are drawn for each boundary pixel and the motion difference between the internal and external parts of object boundaries (supposed to be, respectively, moving and static), Boundary Motion Contrast, is computed as shown in the following equation:

$$\begin{aligned} BMC(t;i) &= \omega_i \left(1 - exp\left(-\frac{\|v_O^i(t) - v_I^i(t)\|}{\sigma^2}\right)\right) \\ \omega_i &= R(v_O^i(t)) \cdot R(v_I^i(t)), \end{aligned} \tag{7.6}$$

where v_O and v_I are the mean motion vectors of the $M \times M$ windows centered in the pixels P_I and P_O for each *i-th* boundary pixel of the foreground region. $R(.)$ represents the reliability of the motion vectors [Erdem et al., 2004a]. The evaluation measures are defined as follows:

$$DM1_{O_j}(t) = \frac{1}{K_t}\sum_{i=1}^{K_t} BMC(t;\, i,\, j), \tag{7.7}$$

$$DM1(t) = \min_j\big(DM1_{O_j}(t)\big), \tag{7.8}$$

where K_t is the total number of contour pixels, BMC is the boundary motion contrast of the *i-th* pixel of the *j-th* foreground region being analyzed, O_j. Its value ranges from 0 (lowest segmentation quality) to 1 (highest segmentation quality). Similarly, this measure could be used to detect correctly segmented boundary pixels if they are above a certain threshold T_2. Thus, a fourth measure is defined as follows:

$$DM2_{O_j}(t) = \frac{\#(BMC(t;\, i) > T_2)}{K_t}, \tag{7.9}$$

$$DM2(t) = \min_j\big(DM2_{O_j}(t)\big). \tag{7.10}$$

Similarly to the BSCC feature, the main advantages of BMC are its low complexity and its possibility to detect failures at finer level (boundary pixel). The only assumption of the measure is that the foreground regions of the sequence have to be moving and can not be static. For this measure, the parameters to study are the normal line length L, the size M of the window around the P_O and P_I points and the threshold, T_2, used in the DM2 measure. As it can be observed the two measures based on motion, DM1 and DM2, fall in the range $[0, 1]$.

7.4 NGT measures for video object tracking quality

We have chosen representative measures for online track quality estimation to compare their performance. We have excluded the ones based on prior models as they need training data, being not adequate for the feedback model proposed in chapter 5. Furthermore, we have not selected any hybrid-based approaches as they are based on the measures from the other categories.

7.4.1 Trajectory-based measures

For this category, two measures have been selected based on forward and reverse analysis of tracking data. As a representative approach of the FM sub-category, we have selected the Motion Smoothness measure (MS) [Wu and Zheng, 2004]. It is defined as:

$$MS(t) = d_E(T(t), T(t-\Delta)) = \sqrt{(x(t) - x(t-\Delta))^2 + (y(t) - y(t-\Delta))^2}, \tag{7.11}$$

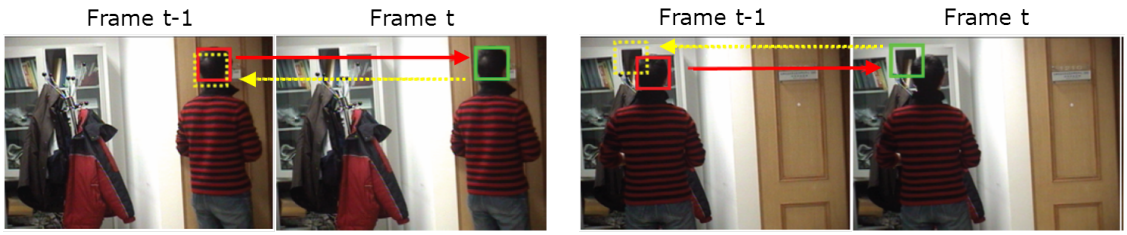
Fig. 7.3. Example of the Motion Smoothness measure (MS) for track quality estimation. Tracking results are represented as green ellipses.

where d_E is the Euclidean distance, $T(t)$ is the target position (at times t and $t-\Delta$; usually consecutive). This measure provides high values when there are relevant changes in the estimated position of the target. It can be considered as a derivative filter of the trajectory distance. Fig. 7.3 shows an example of the score obtained by the MS measure.

As a representative approach of the RM sub-category, we have selected the Template Inverse Matching measure (TIM) [Liu et al., 2008]. The mean-shift algorithm is used for computing the tracking data in forward direction. Then, the template matching is applied for reverse tracking (i.e., backward direction). For every frame t, the measure is defined as:

$$TIM(t) = d(T^F(t-1), T^B(t-1)) = \sqrt{\left(\frac{x^F(t-1) - x^B(t-1)}{W^F(t-1)}\right) + \left(\frac{y^F(t-1) - y^B(t-1)}{H^F(t-1)}\right)}, \tag{7.12}$$

where the super-indexes F and B represent, respectively, the forward and reverse directions of the target position $T(t)$ at time $t-1$; (x, y), H and W are, respectively, the center coordinate, the height and the width of the target. Unlike MS, TIM is robust against fast position changes in case of successful tracking because the change can be recovered by the backward tracking.



(a) (b)

Fig. 7.4. Example of the Template Inverse Matching measure (TIM) for (a) correct and (b) wrong tracking (extracted from [Liu et al., 2008]). Tracking results, previous target position and reverse tracking estimation are represented as, respectively, green, red and yellow rectangles.

Fig. 7.4 illustrates an example of the TIM measure.

7.4.2 Feature-based measures

For this category, two non-model based measures have been selected. They analyze the distributions provided for the target state and the likelihood of the observation. A prior distribution is computed for the successful track status and a (posterior) distribution is obtained for each time t. Then, the two distributions are compared to get the quality measure. Firstly, [Vaswani, 2007] computes the negative log-likelihood of the current observation (OL) and it is approximated as follows:

$$OL(t) \approx -log\left(\frac{1}{N}\sum_{i=1}^{N} w_i(t)\right), \tag{7.13}$$

where $\omega_i(t)$ is the i-th sample of the observation likelihood distribution at time t and N is the number of samples. OL is useful for measuring quick changes in the observation likelihood of the tracked target. However, its performance decreases for slow changes or in presence of distractors (i.e., objects similar to the target). Fig. 7.5 shows an example of the OL measure.

Moreover, [Badrinarayanan et al., 2007b] analyzes the covariance of the target state distribution before and after weighting by the observation likelihood. It is defined as:

$$COV(t) = \frac{det\,[C_b]}{det\,[C_a]}, \tag{7.14}$$

where C_b and C_a are, respectively, the covariance matrix of the state distribution before and after weighting by the observation likelihood (assuming equal weights for the computation of C_b). The value of COV increases when the distribution uncertainty grows (in terms of variance) and tends to zero as the state distribution becomes more peaked. Fig. 7.6 depicts an example of the COV measure.

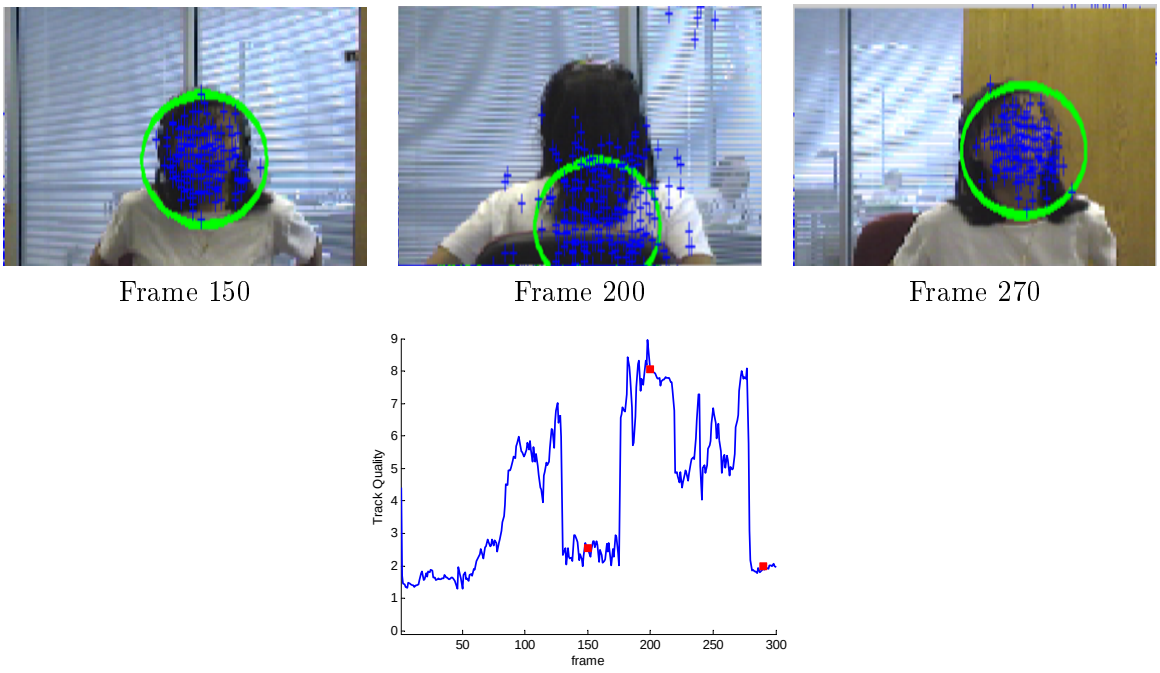


Fig. 7.5. Example of the Observation Likelihood measure (OL) for track quality estimation. Tracking results and the spatial localization of particles are represented as, respectively, green ellipses and blue crosses.

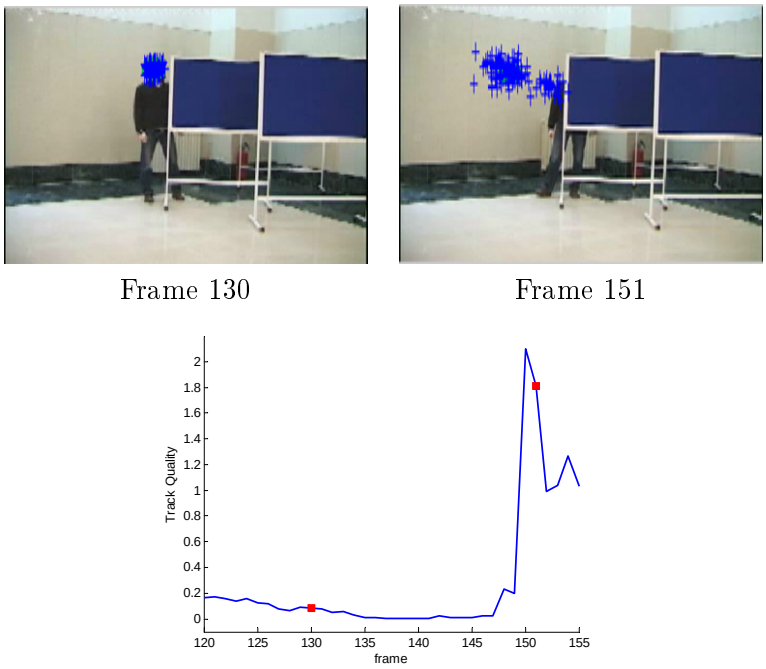


Fig. 7.6. Example of the state covariance measure (COV) for track quality estimation. Tracking results and the spatial localization of particles are represented as, respectively, green ellipses and blue crosses.

7.5 Evaluation methodology

We propose a methodology to evaluate the selected NGT measures. It studies their similarity to GT measures and considers the use of appropriate GT measures, test sequences and algorithms to estimate their quality. This section describes them for the segmentation and tracking tasks.

7.5.1 Foreground segmentation quality

7.5.1.1 Ground-truth measures

As GT measures, we use the Precision for foreground detection ($P1$). The Recall of foreground ($R1$) as well as the Precision and Recall of background segmentation ($P0$ and $R0$) have been excluded due to the NGT measures are not able to evaluate missed foreground detections ($R1$) or background segmentation results ($P0$ and $R0$). The $P1$ measure is defined as follows:

$$P1 = \frac{TP}{TP + FP}, \tag{7.15}$$

where TP and FP indicate, respectively, the number of foreground pixels correctly and wrongly detected. $P1$ ranges from 0 to 1 if, respectively, none or all foreground pixels are detected with respect to the annotations. Then, we use the Pearson product-moment correlation coefficient [Chen and Popovich, 2002] between the $P1$ measure and the NGT quality estimators:

$$\rho = \left| \frac{E\left[(X^e - \mu_{X^e})(X^g - \mu_{X^g})\right]}{\sigma_e \sigma_g} \right|, \tag{7.16}$$

where X^e and X^g are, respectively, the scores of the NGT and GT measures for each video sequence; μ_{X^e} and μ_{X^g} are their respective means and σ_{X^e} and σ_{X^g} are their respective standard deviations; $\rho \in [0, 1]$ with values close to 1 (0) indicating high (low) GT correlation.

7.5.1.2 Dataset description

The test sequences have been selected from the CVSG dataset[2] that has been designed for the evaluation of moving object segmentation, allowing to combine different real foregrounds and backgrounds. It is composed of high-quality uncompressed video sequences of size 720x576 (and associated ground-truth) classified according to several criteria. We propose to use some of these criteria to evaluate the performance of the NGT measures under different complexities of the background and the foreground data. A description of the selected sequences and their main characteristics are shown in Table 7.3. The sequences with camera motion have been excluded for the experiments because the evaluated segmentation algorithms do not handle them. Sample frames of these sequences are shown in Fig. 7.7.

[2] http://www-vpu.eps.uam.es/CVSG/

Fig. 7.7. Sample frames from the segmentation evaluation dataset (CVSG). From top-left to bottom-right: ID01, ID04, ID05, ID06, ID07, ID08, ID09, ID10 and ID11.

Seq	Length	Background		Foreground	
	Frames	Textures	Multimodal	Velocity	Size
ID01	750	High	Low	Low	Medium
ID04	1250	Low	Low	Low	Medium
ID05	752	Medium	High	Low	Medium
ID06	672	Medium	High	Low	Medium
ID07	620	Medium	High	Low	Medium
ID08	794	Medium	Medium	Low	Medium
ID09	1380	Medium	Medium	High	Medium
ID10	307	Medium	Medium	High	Medium
ID11	732	Low	Low	High	Low

Table 7.3: Description of the segmentation evaluation dataset.

7.5.1.3 Selected segmentation algorithms

As segmentation algorithms, we have selected four representative approaches of the *background subtraction* technique commonly used in video-surveillance: the first two [Stauffer and Grimson, 1999; Elgammal et al., 2000] independently model each background pixel whereas the other two [Oliver et al., 2000; Cavallaro et al., 2005] consider spatial relations of neighborhood pixels. Additionally, a noise filter has been applied to remove the salt&pepper noise of the obtained foreground masks by using mathematical morphology [Salembier and Ruiz, 2002]. Parameter tuning of each approach has been done according to the results reported in [Herrero and Bescos, 2009]. The first approach is the Mixture of Gaussians (MoG) [Stauffer and Grimson, 1999] where the movement of each background pixel is represented with a set of weighted Gaussian distributions. The distributions with higher weights are considered to model the background; the remaining to model the foreground. Foreground pixel detection is decided if the pixel does not fall into the deviation around the mean of any of the Gaussians that model the background. This approach is useful to analyze sequences with multimodal background. The second approach [Elgammal et al., 2000] is the Kernel Density Estimator (KDE) that estimates the probability density function of each pixel by using the last N frames allowing to analyze multimodal backgrounds. Foreground or Background pixel detection is decided if its likelihood of belonging to the estimated pixel density is, respectively, lower or higher than a predefined threshold. The third approach [Cavallaro et al., 2005] is the Gamma method (GAMMA). It is based on a pixel neighborhood analysis by subtracting a square window between the current and background images (around each pixel). This subtraction is modeled as a Chi-square distribution assuming a Gaussian noise that affects each pixel. The final decision is taken by thresholding the probability of belonging to the Chi-square distribution. This approach eliminates the salt&pepper noise in the foreground binary mask. The fourth approach is the Eigenbackgrounds (EigBG) [Oliver et al., 2000]. It is based on applying principal component analysis to the previous N frames to capture the spatial relations. A set of basis functions is obtained as a result and each new frame is projected into the eigenspace defined by these functions to remove foreground objects. Foreground detection is obtained by comparing each frame with its back projection.

7.5.2 Video object tracking quality

7.5.2.1 Ground-truth measures

For evaluating the NGT measures, we analyze their performance in global terms and at the start/end of a failure. For the global analysis, we use ROC analysis evaluating the discrimination of their values between two classes: successful and unsuccessful tracks. An unsuccessful track is determined if the Displacement Error Rate (DER) [Han et al., 2008] is above a certain value that depends on the minimum allowed overlap between the GT data and the detected areas (e.g.,

Dataset	Targets	Size	Characteristics
SPEVI	H1	320x240	SC, C, O
CLEMSON	H2 – H5	128x96	SC, AC, C, O
PETS01	P1 – P4	768x576	SC, O
CAVIAR	P5	384x288	C
VISOR	P6, P7	352x288	SC, O

Table 7.4: Description of the tracking evaluation dataset (SC: Scale changes. AC: Appearance changes. IC: Illumination changes. O: occlusions. C: clutter.).

a DER value of $\sqrt{2}/2$ indicates an overlap of 50%). DER computes the distance between the target estimation, T_e, and the GT annotation, T_{GT}, as follows:

$$DER = \frac{d_E(T_e, T_{GT})}{\sqrt{A_{GT}}} = \frac{\sqrt{(x_e - x_{GT})^2 + (y_e - y_{GT})^2}}{\sqrt{A_{GT}}}, \tag{7.17}$$

where $(x, y)_e$, $(x, y)_{GT}$ are, respectively, the center location of the target estimated and annotated; and A_{GT} is the area of the GT annotation. Additionally, the accuracy of the DER measure has been verified by visually defining the time instants when the tracker fails.

7.5.2.2 Dataset description

The evaluation dataset is composed of sequences from the SPEVI[3] and CLEMSON[4] public datasets for face tracking, the PETS2001 dataset[5], the CAVIAR dataset[6] and the VISOR dataset[7]. The selected sequences represent common issues for tracking analysis in real scenarios such as occlusions, scale and appearance changes and clutter. Their characteristics are summarized in Table 7.4 and the target initialization is shown in Fig. 7.8.

7.5.2.3 Selected tracking algorithm

As tracking algorithm, we have selected the particle filter framework due to the huge amount of related literature and, therefore, the potential use of the obtained conclusions. In this framework, the output of an algorithm at each time step can be represented by a weighted sample set $X_t = \left\{(\mathbf{x}_t^{(n)}, \pi_t^{(n)}\right\}_{n=1....N}$. Each sample (or particle) $\mathbf{x}_t^{(n)}$ represents one hypothetical state of the object with a corresponding discrete sampling probability $\pi_t^{(n)}$, where $\sum_{n=1}^{N} \pi_t^{(n)} = 1$. This sample set X is used to estimate track quality with the selected NGT measures.

[3] ftp://motinas.elec.qmul.ac.uk/pub/single_face
[4] http://www.ces.clemson.edu/~stb/research/headtracker
[5] http://www.cvg.rdg.ac.uk/PETS2001/
[6] http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/
[7] http://imagelab.ing.unimore.it/visor/

Fig. 7.8. Target initialization for the tracking evaluation dataset. From top-left to bottom-right: Faces: H1, H2, H3, H4 and H5 and Pedestrians: P1, P2, P3, P4, P5, P6 and P7.

Specifically, we have chosen the color-based particle filter approach described in [Nummiaro et al., 2003]. The target is represented as an ellipse and each sample is defined as:

$$\mathbf{x} = \{x,\, y,\, H_x,\, H_y,\, \theta\}\,, \tag{7.18}$$

where $(x,\, y)$ specify the center of the ellipse; H_x, H_y the half of axes length; and θ the ellipse rotation. A zero-order motion model is used to propagate the sample set:

$$X_t = X_{t-1} + W_{t-1}, \tag{7.19}$$

where W_{t-1} is a multivariate Gaussian random variable. To weight the sample set, the Bhattacharyya coefficient is computed between the target and hypotheses histograms.

Finally, the target position estimation is given by the mean state of the samples defined as:

$$E[X_t] = \sum_{n=1}^{N} \mathbf{x}_t^{(n)} \pi_t^{(n)}, \tag{7.20}$$

where N is the number of samples; $x_t^{(n)}$ and $\pi_t^{(n)}$ correspond to the state estimation; and the weight associated to each sample of the particle filter.

The tracker configuration was the same for all the experiments. Color histograms were created using 8x8x8 bins in the RGB and HSV color spaces for, respectively, pedestrian and face targets; its parameters were heuristically set to $\sigma_{x,y} = 6$, $\sigma_{H_x,H_y} = 2$, $\sigma_\theta = 4°$, $\sigma_c = 0.2$ and $N = 500$.

7.6 Optimum parameter selection for quality estimators

A study of the optimum parameters has been carried out for the selected NGT measures to evaluate foreground segmentation and tracking. In this section, we describe them.

7.6.1 NGT measures for foreground segmentation quality

For the NGT measures for foreground segmentation quality (DC1, DC2, DM1 and DM2), we have determined the optimum parameters for the different segmentation results obtained with the selected algorithms. The optimization process has been divided in two stages using sequences with unimodal backgrounds from the CVSG dataset in order to avoid the adaptation of the parameter values to the high amount of segmentation errors found in multimodal sequences. A summary of the evaluation results is depicted in Fig. 7.9.

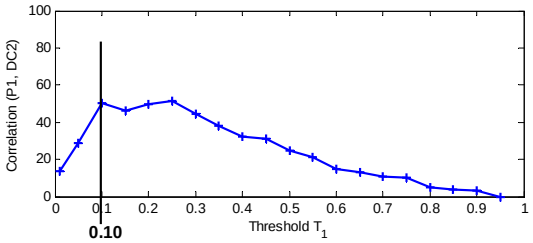
Firstly, we have performed an exhaustive search of the optimum L and M parameter values by using the DC1 and DM1 measures. The optimum selection criterion is the maximum correlation between DC1 and DM1 with the GT measure $P1$. As it can observed in Fig. 7.9(a) and 7.9(c), the optimum values (maximum correlation) for the DC1 measure was 0.732 for the parameters $L = 5$ and $D = 3$ whilst for the DM1 measure was 0.451 for the parameters $L = 5$ and $D = 3$. Then, the optimum values of the thresholds T_1 and T_2 (used in measures DC2 and DM2) are calculated by applying an exhaustive search and considering the optimum values of L and M for each measure. Results are shown in Fig. 7.9(b) and 7.9(d). The optimum selection criterion is the maximum correlation between the DC2 and DM2 with the GT measure $P1$. Finally, the optimum values were $T_1 = 0.10$ and $T_2 = 0.25$ for the DC2 and DM2 measures.

M	L				
	1	3	5	7	9
1	65.3±2.6	68.3±3.2	65.3±3.3	62.4±4.3	59.8±12.0
3	-	72.7±4.1	**73.2±4.3**	69.3±7.3	66.8±8.2
5	-	-	72.8±3.3	70.0±2.6	67.1±7.5
7	-	-	-	71.2±4.3	67.1±5.0
9	-	-	-	-	66.1±10.2

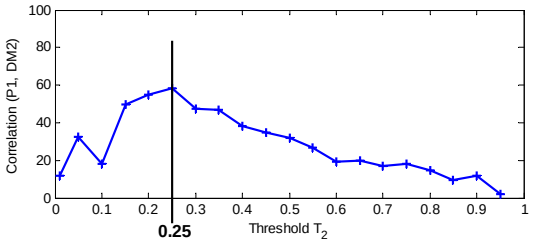
(a)

M	L				
	1	3	5	7	9
1	25.0±15.4	29.0±12.8	31.1±8.3	30.0±12.4	29.1±9.8
3	-	36.3±5.6	**45.1±4.6**	35.2±9.4	34.0±10.3
5	-	-	39.3±7.7	36.6±8.3	36.9±6.9
7	-	-	-	31.5±7.7	38.3±5.0
9	-	-	-	-	32.5±9.4

(c)



(b)



(d)

Fig. 7.9. Optimum parameter estimation for NGT segmentation measures. Results are the mean of the correlation obtained by the four selected algorithms (in percentage) of (a) DC1, (b) DM1, (c) DC2 and (d) DM2 with P1 measure. Maximum values are bold-marked.

7.6.2 NGT measures for video object tracking quality

For the NGT track quality measures (MS, TIM, OL and COV), the estimation of the optimum parameters is a complex task being application dependent in most of the cases. Therefore, a study similar to the one performed for foreground segmentation has no sense and we have decided to use the parameters proposed by the corresponding authors. For the MS measure, we use $\Delta = 1$. The TIM measure does not have any parameters to set. For the Feature-based measures (OL and COV), the unique available parameter is the number of samples of the distribution likelihood (N). In the particle filter framework, it corresponds to the number of particles (set to $N = 500$).

7.7 Experimental results

We carried out an evaluation of the performance of the selected NGT measures. This section shows and discusses the obtained results following the previously described methodology.

7.7.1 Foreground segmentation quality

We have studied the performance of the NGT segmentation measures under different background and foreground characteristics (defined in Table 7.3). For each one, appropriate test sequences are selected and the segmentation algorithms are applied to them. Then, we evaluate their results with the NGT and GT measures and analyze in which cases the NGT measures are accurate.

Algorithm	GT	NGT Color				NGT Motion			
	P1	DC1		DC2		DM1		DM2	
	Mean	Mean	ρ	Mean	ρ	Mean	ρ	Mean	ρ
MoG	96.5	**19.9**	**.72**	75.7	.52	15.3	.48	70.3	.45
KDE	88.4	**18.4**	**.73**	70.2	.60	10.9	.35	68.9	.34
GAMMA	93.3	**19.7**	**.74**	74.3	.55	9.9	.41	70.4	.40
EigBG	84.3	**13.6**	**.75**	46.6	.52	7.6	.49	55.6	.48

(a)

Algorithm	GT	NGT Color				NGT Motion			
	P1	DC1		DC2		DM1		DM2	
	Mean	Mean	ρ	Mean	ρ	Mean	ρ	Mean	ρ
MoG	52.5	25.3	.20	73.4	.19	**16.3**	**.21**	77.5	.09
KDE	33.4	**18.4**	**.18**	55.1	.16	11.5	.15	69.5	.11
GAMMA	50.3	19.7	.13	**66.2**	**.17**	8.0	.14	80.3	.05
EigBG	47.1	**13.6**	**.18**	69.4	.11	19.6	.09	99.3	.03

(b)

Table 7.5: Performance of NGT segmentation evaluation for (a) low and (b) high background motion. For each measure, results are reported as the mean value and the correlation with P1 (ρ). Best results for each algorithm are marked in bold. (MoG [Stauffer and Grimson, 1999]; KDE [Elgammal et al., 2000]; GAMMA [Cavallaro et al., 2005]; EigBG [Oliver et al., 2000]).

7.7.1.1 Background motion

Background motion is a key issue in foreground segmentation because it is difficult to handle. The test sequences ID1, ID4 and ID11 were used for computing the results of low background motion and the remaining ones for medium-high background motion. The obtained results are summarized in Table 7.5. As we can observe, high background motion reduces the accuracy of the segmentation algorithms increasing the number of wrong detections (P1 measure). For the color-based measures, the observed performance decrease (correlation values) is caused by the wrong detected objects (moving background) as they have color-boundaries. For the motion-based measures, the background motion also produces motion boundaries and the measures wrongly evaluate the segmented background as good so their correlation values are heavily reduced.

7.7.1.2 Background texture

Different background textures were tested for NGT evaluation. Test sequences ID4 and ID1 were used as low and high textured backgrounds. The obtained results are summarized in Table 7.6. As we can observe, high-textured backgrounds increase the performance (correlation) of motion-based NGT measures and slightly decrease the performance of color-based NGT measures because the motion boundaries are less difficult to estimate, whilst color boundaries are more difficult to obtain. On the contrary, the color-based NGT measures are more suitable for low textured backgrounds because color boundaries are easier to estimate and decrease the performance of motion-based NGT measures because the extraction of motion boundaries is more difficult.

Algorithm	GT	NGT Color				NGT Motion			
	P1	DC1		DC2		DM1		DM2	
	Mean	Mean	ρ	Mean	ρ	Mean	ρ	Mean	ρ
MoG	99.9	**19.0**	**.78**	78.7	.60	9.3	.41	65.3	.40
KDE	92.1	**17.0**	**.75**	75.4	.48	8.8	.53	69.4	.40
GAMMA	95.4	**18.2**	**.71**	77.6	.62	12.0	.51	70.9	.38
EigBG	80.5	11.0	.70	**81.3**	**.88**	9.9	.49	47.4	.41

(a)

Algorithm	GT	NGT Color				NGT Motion			
	P1	DC1		DC2		DM1		DM2	
	Mean	Mean	ρ	Mean	ρ	Mean	ρ	Mean	ρ
MoG	96.8	18.8	.55	74.9	.49	**13.3**	**.55**	67.4	.42
KDE	90.6	**16.3**	**.63**	60.5	.35	12.8	.56	73.2	.45
GAMMA	94.3	**18.9**	**.59**	71.9	.56	15.4	.54	75.1	.39
EigBG	82.9	19.2	.61	**72.3**	**.70**	20.4	.50	59.9	.44

(b)

Table 7.6: Performance of NGT segmentation evaluation for (a) low and (b) high background texture. For each measure, results are reported as the mean value and the correlation with P1 (ρ). Best results for each algorithm are marked in bold. (MoG [Stauffer and Grimson, 1999]; KDE [Elgammal et al., 2000]; GAMMA [Cavallaro et al., 2005]; EigBG [Oliver et al., 2000]).

7.7.1.3 Foreground velocity

Different velocities of foreground objects have been tested to study the performance of the NGT measures. The test sequence ID4 was used for low foreground velocity, whilst sequence ID11 was used for high foreground velocity. The obtained results are summarized in Table 7.7. As we can observe, the velocity of the foreground objects does not significantly affect the color-based NGT measures, whilst the motion-based NGT measures present a dependence on the object velocity. The computation of the motion boundaries has to be adapted to the specific characteristics of each sequence by setting the optimum parameter for the motion vector calculation (e.g., block size and area search for block matching).

7.7.1.4 Foreground size

Different foreground object sizes have been tested to study the performance of the NGT measures. All test sequences have been used and the obtained results were classified depending on the object size considering small size if it has less than 200 pixels and large size otherwise. The obtained results are summarized in Table 7.8. As we can observe, the size of the object does not affect the color and motion-based NGT. However, the NGT measures will not work with very thin objects (e.g., walking sticks) because the parameter L could be higher than the dimensions of the object (width or height). The low accuracy (correlation) of NGT measures is because the presence of sequences with multimodal backgrounds produces wrong objects multiple sizes.

Algorithm	GT	NGT Color				NGT Motion			
	P1	DC1		DC2		DM1		DM2	
	Mean	Mean	ρ	Mean	ρ	Mean	ρ	Mean	ρ
MoG	99.9	**19.0**	**.78**	78.7	.60	9.3	.41	65.3	.41
KDE	92.1	**17.0**	**.75**	75.4	.48	8.8	.53	69.4	.40
GAMMA	95.4	**18.2**	**.71**	77.6	.62	12.0	.51	70.9	.38
EigBG	80.5	11.0	.70	**81.3**	**.88**	9.9	.49	47.4	.41

(a)

Algorithm	GT	NGT Color				NGT Motion			
	P1	DC1		DC2		DM1		DM2	
	Mean	Mean	ρ	Mean	ρ	Mean	ρ	Mean	ρ
MoG	90.3	**19.0**	**.75**	74.0	.49	10.1	.45	63.1	.37
KDE	89.6	**17.3**	**.72**	71.5	.42	9.9	.36	70.1	.31
GAMMA	85.9	**15.9**	**.68**	73.1	.59	11.5	.50	66.5	.35
EigBG	95.3	20.2	.64	**70.1**	**.69**	25.0	.39	69.6	.45

(b)

Table 7.7: Performance of NGT segmentation evaluation for (a) low and (b) high foreground velocity. For each measure, results are reported as the mean value and the correlation with P1 (ρ). Best results for each algorithm are marked in bold. (MoG [Stauffer and Grimson, 1999]; KDE [Elgammal et al., 2000]; GAMMA [Cavallaro et al., 2005]; EigBG [Oliver et al., 2000]).

Algorithm	GT	NGT Color				NGT Motion			
	P1	DC1		DC2		DM1		DM2	
	Mean	Mean	ρ	Mean	ρ	Mean	ρ	Mean	ρ
MoG	70.5	**17.7**	**.30**	81.1	.29	17.0	.21	78.1	.19
KDE	60.8	**12.4**	**.28**	69.9	.27	12.6	.18	69.5	.12
GAMMA	71.2	**17.4**	**.35**	81.2	.32	13.5	.20	80.3	.18
EigBG	57.3	**12.5**	**.31**	66.7	.30	10.6	.17	88.1	.11

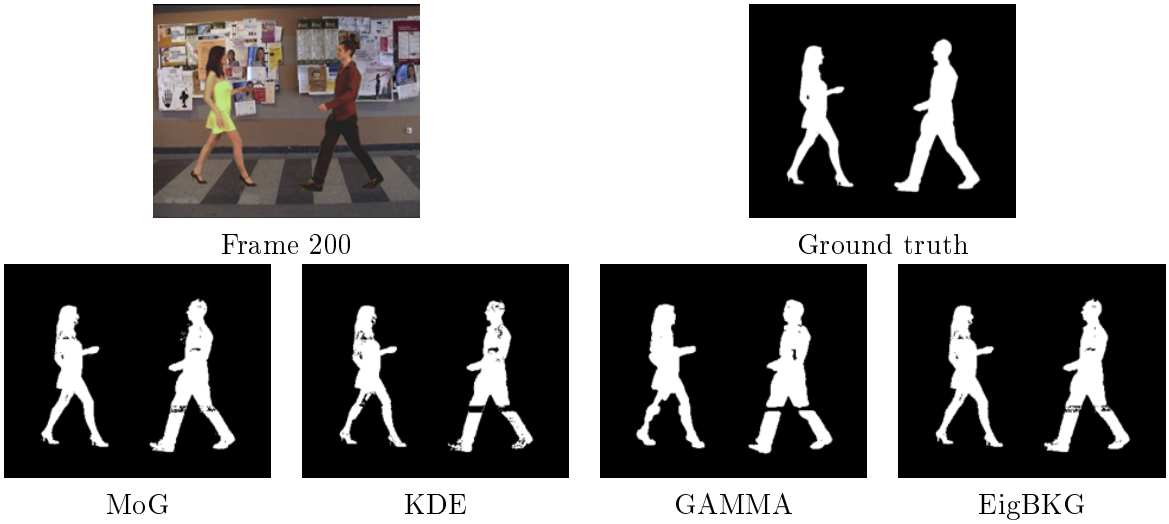
(a)

Algorithm	GT	NGT Color				NGT Motion			
	P1	DC1		DC2		DM1		DM2	
		Mean	ρ	Mean	ρ	Mean	ρ	Mean	ρ
MoG	44.4	**21.8**	**.26**	90.1	.11	21.0	.10	88.1	.10
KDE	25.1	**14.0**	**.19**	76.9	.12	16.6	.08	77.5	.08
GAMMA	49.8	**18.9**	**.29**	83.0	.25	18.5	.13	83.8	.11
EigBG	43.3	**15.6**	**.21**	52.9	.13	13.6	.15	89.9	.04

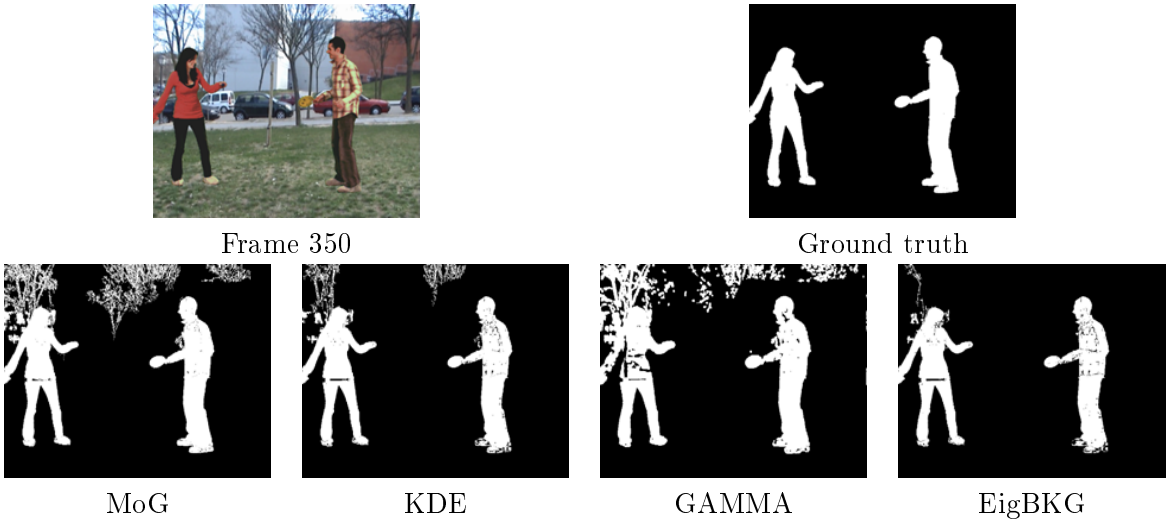
(b)

Table 7.8: Performance of NGT segmentation evaluation for (a) low and (b) high foreground size. For each measure, results are reported as the mean value and the correlation with P1 (ρ). Best results for each algorithm are marked in bold. (MoG [Stauffer and Grimson, 1999]; KDE [Elgammal et al., 2000]; GAMMA [Cavallaro et al., 2005]; EigBG [Oliver et al., 2000]).

An example is depicted in Fig. 7.10. The variation of the DC1 measure is proportional to the P1 values of each algorithm (except the GAMMA one in the case show in Fig. 7.10(a)). The DC2 measure has a performance decrease with respect to DC1. On the other hand, motion-based measures (DM1 and DM2) do not present enough correlation with the P1 measure.



(a)



(b)

Algo.	GT	NGT			
	P1	DC1	DC2	DM1	DM2
MoG	99.8	17.9	83.5	22.1	85.3
KDE	88.3	15.3	77.3	18.5	80.1
GAMMA	91.5	14.7	68.5	17.0	76.6
EigBKG	95.4	19.0	74.5	19.3	79.4

(c)

Algo.	GT	NGT			
	P1	DC1	DC2	DM1	DM2
MoG	71.6	20.3	67.5	12.3	66.6
KDE	77.3	23.3	73.4	18.4	77.9
GAMMA	68.9	19.8	61.7	22.3	83.1
EigBKG	88.2	26.1	77.0	21.5	81.8

(d)

Fig. 7.10. Examples for estimation of foreground segmentation quality. Data corresponds to the analysis of the frame 200 of ID1 test sequence ((a) the segmentation mask and (c) its evaluation) and to the frame 350 of ID5 test sequence ((b) the segmentation mask and (d) its evaluation). (MoG [Stauffer and Grimson, 1999]; KDE [Elgammal et al., 2000]; GAMMA [Cavallaro et al., 2005]; EigBG [Oliver et al., 2000]).

Measure	AUC	FPR	TPR
MS	.55 ± .0599	.43 ± .0795	.53 ± .0727
TIM	.69 ± .0358	.37 ± .0481	.60 ± .0651
OL	.78 ± .0887	.20 ± .0554	.65 ± .1133
COV	.70 ± .0675	.35 ± .0619	.72 ± .0986

Table 7.9: ROC results for NGT tracking evaluation (mean ± standard deviation of 10 runs). (AUC: Area Under Curve, FPR: False Positive Rate, TPR: True Positive Rate; MS: Motion Smoothness [Wu and Zheng, 2004]; OL: Observation Likelihood [Vaswani, 2007]; COV: State Covariance [Badrinarayanan et al., 2007b]; TIM: Template Inverse Matching [Liu et al., 2008]).

7.7.2 Video object tracking quality

For video object tracking quality, we present an overall comparison of the selected NGT measures as well as a detailed analysis of the performance of the quality measures for a specific sequence.

7.7.2.1 Overall quality comparison

Due to the statistically nature of the selected tracking algorithm, several runs have been performed to get meaningful results. A summary of the experiments is shown in Table 7.9 and Fig. 7.11. Feature-based measures (OL and COV) performed better than trajectory-based measures (MS and TIM) because the tested sequences contained failures that produced a change in the state or observation likelihood and sometimes a fast position change of the target. Additionally, feature-based measures presented high standard deviation showing their dependency on the probabilistic analyzed data. Among the trajectory-based measures, TIM presented the highest area under the curve (AUC) value due to the robustness introduced by the backward tracking stage. MS had low accuracy and depends on the movement of the target. Among the feature-based measures, OL outperformed COV as the majority of the evaluated failure types implied a change in the target observation likelihood (e.g., particle weights in a particle filter framework) instead of a variation of the state variance.

7.7.2.2 Analysis of the performance of the quality measures

A detailed analysis of the results in the test sequences has been done to evaluate the DER and the selected NGT track quality measures. Fig. 7.12 shows an example of the DER and the track quality measurements for the H5 target of the *seq_mb* sequence. Fig. 7.12(a) shows that DER produced a good measurement of the track quality useful for the overall quality comparison. Regarding the quality of the measures, TIM provided high values at the starting/ending failure frames when the position change was due to model target dissimilarities (Fig. 7.12(d) and 7.12(e)) and similarities with clutter (Fig. 7.12(c)). During the failures, TIM did not provide high values because there was not a position change, whilst outside the failures its lower values indicated

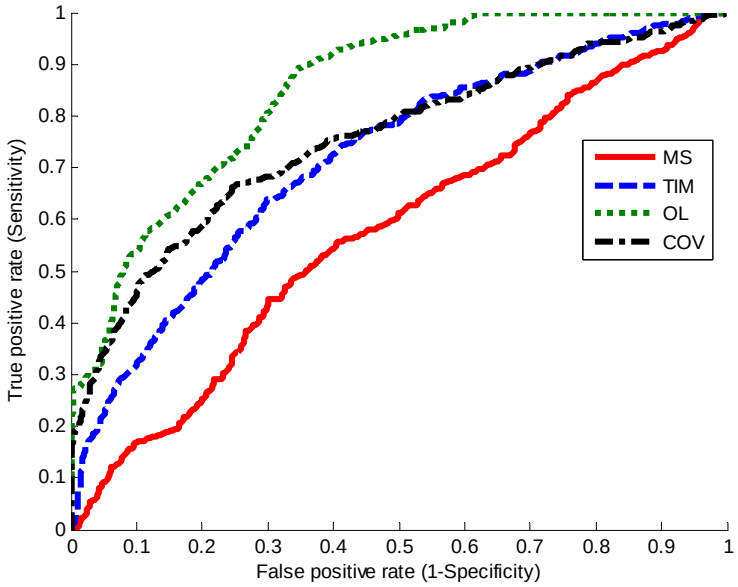


Fig. 7.11. ROC curves for the NGT tracking evaluation. (MS: Motion Smoothness [Wu and Zheng, 2004]; OL: Observation Likelihood [Vaswani, 2007]; COV: State Covariance [Badrinarayanan et al., 2007b]; TIM: Template Inverse Matching [Liu et al., 2008]).

high-quality measurements. MS provided high values during the failures (starting/ending and duration) but also presented high values outside them. In general, MS obtained low performance to measure track quality. OL provided high values during the tracking failures because there was a change in observation likelihood in all the failures. At the starting/ending frames, its results depended on the rate of the observation likelihood change as demonstrated for the slow (Fig. 7.12(e)) and medium (Fig. 7.12(c)) changes. COV obtained good quality measurements in case of occlusion and appearance changes because the state distribution variance is increased to search the lost target. Moreover, the performance of COV is reduced when the target model is adapted to the wrong track (frames 220-260 of Fig. 7.12(d)).

Additional examples of the NGT tracking measures are illustrated in Fig. 7.13. An example of target model similarities is shown in Fig. 7.13(a) in case of clutter. The small change in the observation likelihood and state distribution variance produced a low quality measurement of track quality by OL and COV. TIM obtained good results because there was a change in position (to the wrong estimated target). MS obtained low quality measurement because the target movement was small. Fig. 7.13(b) and 7.13(c) depict examples of target model dissimilarities for an occlusion and an appearance change. The observation likelihood and state distribution variance changes were measured, respectively, by OL and COV obtaining high quality measurements. TIM obtained poor results because the backward tracking used the wrong tracked target as template being only adequate its use for the instant when the dissimilarities were produced. MS also obtained high quality measures because the quick changes of target movement.

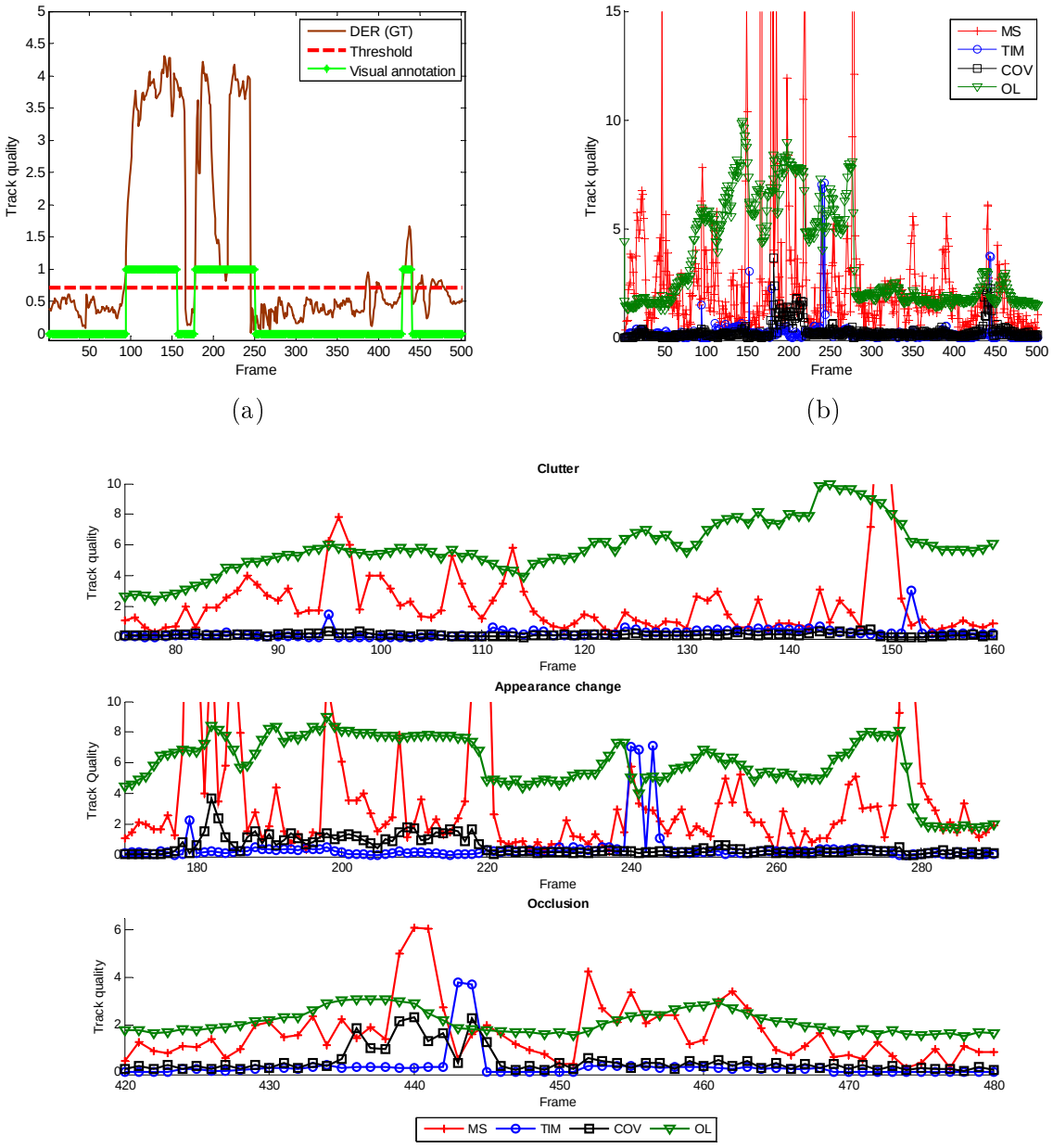


Fig. 7.12. Results for the tracking analysis of H5 target of the "seq_mb" test sequence in terms of (a) Displacement Error Rate (DER), (b) Track quality measures and (c) zoom of the segments of interest for the clutter (frames 90-155), appearance change (frames 170-240) and occlusion (frames 435-445) failures. (MS: Motion Smoothness [Wu and Zheng, 2004]; OL: Observation Likelihood [Vaswani, 2007]; COV: Target state Covariance [Badrinarayanan et al., 2007b]; TIM: Template Inverse Matching [Liu et al., 2008]).

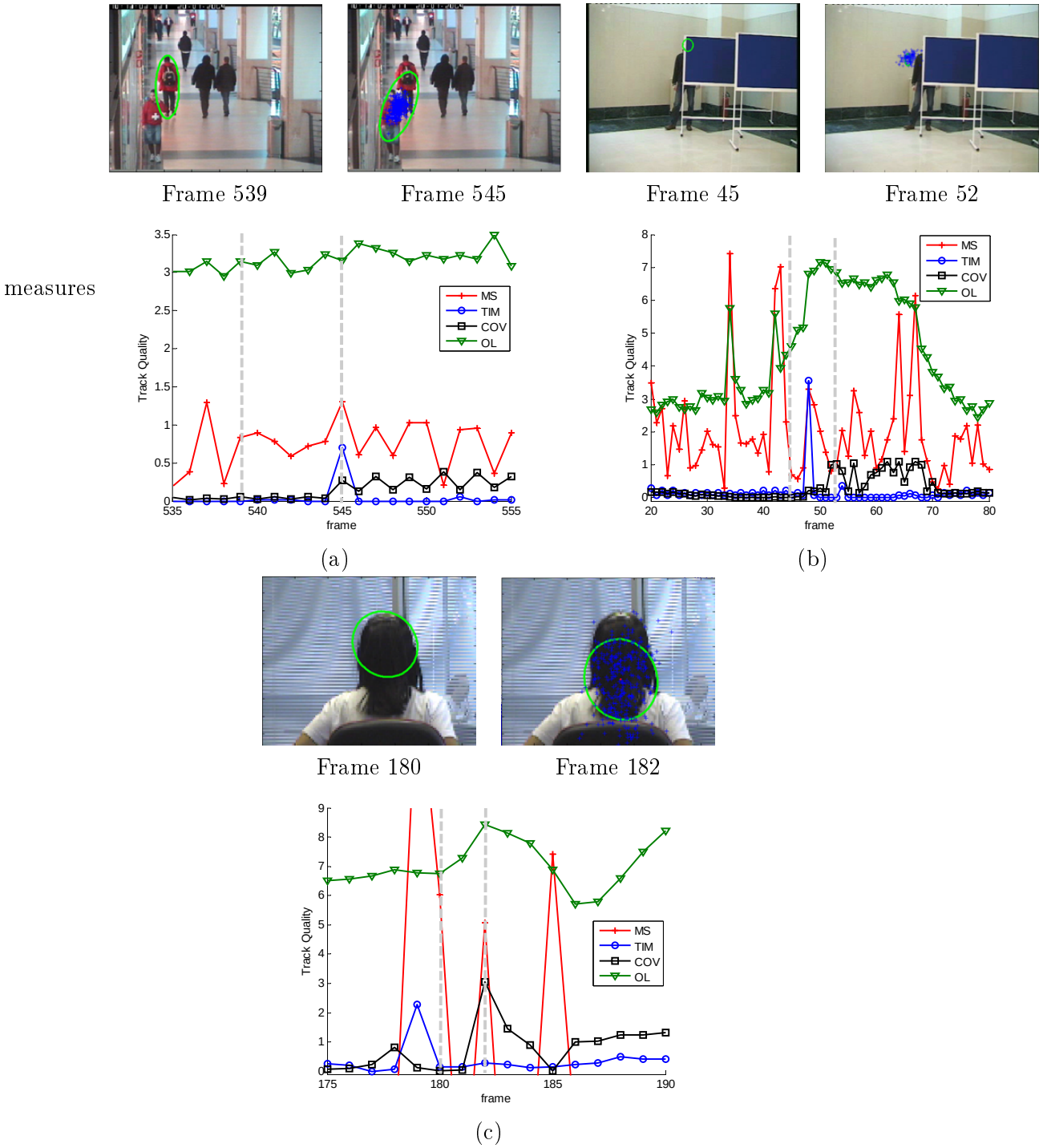


Fig. 7.13. Examples of track quality measures for (a) clutter (target P5), (b) an occlusion (target P6) and (c) appearance change (target H5). Results correspond to ground-truth (top-left), tracking results (particles) (top-right) and the measures (bottom). (MS: Motion Smoothness [Wu and Zheng, 2004]; OL: Observation Likelihood [Vaswani, 2007]; COV: Target state Covariance [Badrinarayanan et al., 2007b]; TIM: Template Inverse Matching [Liu et al., 2008]).

7.8 Summary and conclusions

In this chapter we have studied the evaluation without ground-truth of the video object segmentation and tracking stages. The effective application of feedback in video analysis (using the model of chapter 5) requires mechanisms to estimate the output quality of the processing stages.

For video object segmentation, we have proposed a taxonomy and we have studied four representative measures. They rely on comparing the boundaries of the segmented objects against the color and motion boundaries of the video sequence. An evaluation of the measures under different sequence characteristics has been carried out for the video-surveillance domain. Their correlation with ground-truth measures has been used to check their performance. Experimental results showed that the color-based measures are more accurate than the motion-based ones because motion boundaries are more difficult to estimate than color boundaries due to homogeneous or slow-moving object regions. Background multimodality dramatically affects their performance, whilst the effect of background textures and foreground velocity/size is less noticeable. Among the color-based measures, DC1 performs better than DC2 because it does not need any thresholding operation. Similarly happens with the DM1 and DM2 motion-based measures. As future work, we will study the use of these measures to detect segmentation failures and to feedback foreground segmentation algorithms to improve their accuracy.

For video object tracking, we have introduced a taxonomy and presented a comparative evaluation of online quality estimation measures. Existing approaches have been classified into three categories and the representative ones have been compared. Experimental results using a heterogeneous dataset demonstrated that different measures should be applied to evaluate the overall performance and the start/end time of failures. For measuring the start and the end of a tracking failure, the TIM measure is outperforming the other measures both for target model similarities (e.g., clutter) and dissimilarities (e.g., appearance changes and occlusions); whereas when considering the overall quality, the OL measure obtained the best results.

In the next chapter, we have decided to focus on the estimation of track quality to improve the limitations exhibited in the comparative. We propose a novel method to estimate the track quality an adaptive reverse tracking approach. The idea behind our approach is to identify the status of the target (e.g., target lost, wrong target, looking for the target) and calculate the track quality with a reverse tracker of variable length.

Chapter 8

On-line video tracker evaluation using adaptive reverse tracking

8.1 Introduction[1]

Video tracking is an important step in many applications, such as video surveillance, human-computer interaction and object-based video compression. Video data present high complexity and variability because of pose changes, illumination variations, occlusions and clutter. Under such conditions, no single video tracker can perform perfectly in all situations and failures are expected in real tracking scenarios. An online track failure detector and quality estimator is needed to measure tracking performance over time. For developing robust approaches in real-world conditions (e.g., by using the feedback model of chapter 5), self-evaluation is required.

As described in chapter 7, common tracking performance evaluations use *empirical discrepancy methods* [Maggio and Cavallaro, 2011] that compare off-line ground-truth (GT) data with the estimated target state. GT data are expensive to produce and usually cover only small temporal segments of test video sequences, thus representing only a small percentage of data variability. This limitation makes it difficult to extrapolate the performance evaluation results to (unlabeled) new sequences. Moreover, evaluation using GT is not feasible for on-line performance analysis [Kasturi et al., 2008]. To extend the applicability of performance evaluation, *empirical standalone methods* (ESM) for track-quality estimation without GT data have been defined for large unlabeled datasets, self-tuning (automatic control via on-line analysis), comparative ranking and tracker fusion. ESMs use data derived from the estimated trajectories, such as motion smoothness [Black et al., 2003], area consistency [Chau et al., 2009], or time-reversibility [Wu et al., 2010]; statistical properties of the tracker output, such as observation likelihood [Vaswani, 2007], spatial uncertainty [Maggio et al., 2007], or consistency checks [Van der Heijen, 2006]; com-

[1] This chapter is based on the publication *"J.C. SanMiguel, A. Cavallaro, and J.M. Martínez. Adaptive on-line performance evaluation of video trackers. submitted to IEEE Trans. on Image Processing, 2010"*.

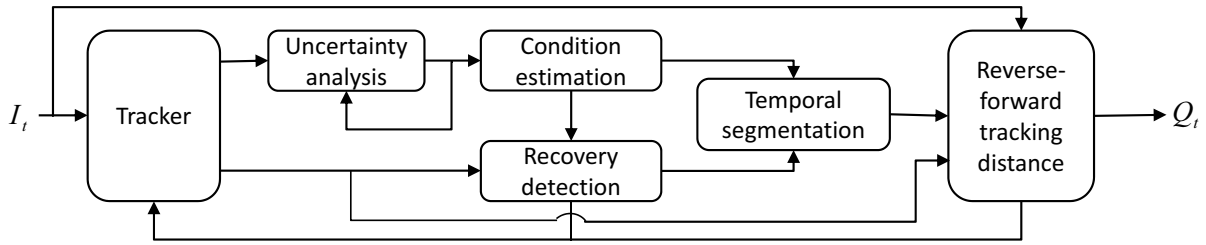


Fig. 8.1. Block diagram of the proposed adaptive on-line evaluator for tracking performance.

plementary features, such as color contrast [Erdem et al., 2004a] or background discriminative power [Collins et al., 2005]; and combinations of these properties [Kalal et al., 2010]. However, they are generally application-dependent [Black et al., 2003; Chau et al., 2009; Erdem et al., 2004a; Collins et al., 2005], non-applicable to long sequences [Wu et al., 2010] or non-adaptive to errors and recoveries of the tracker [Vaswani, 2007; Maggio et al., 2007]. In addition, their use and experimental validation is generally limited to short or low-complexity videos.

To overcome the above mentioned limitations, we propose a novel adaptive empirical stand-alone method for track-quality estimation that is applicable to image sequences with multiple tracking errors and recoveries. The proposed framework is based on a two-stage adaptive strategy that first determines when a target is being successfully tracked (temporal segmentation) and then estimates track quality during successful tracking. The framework effectively combines the filter uncertainty and the time-reversibility constraint of a tracker to measure the quality of the estimated target state. The analysis of the filter uncertainty allows one to detect unstable tracking data and the detection of a recovery after a tracking failure by applying a reverse tracker. We demonstrate the proposed approach in a particle filter framework over a heterogeneous dataset with sequences containing tracking challenges such as occlusions, clutter and appearance changes. A block diagram of the proposed framework is shown in Fig. 8.1.

This chapter is organized as follows. Section 8.2 describes the identification of the target condition, whilst Section 8.3 introduces the track quality estimation. Section 8.4 discusses the results and comparisons with alternative approaches. Finally, Section 8.5 summarizes the chapter.

8.2 Is the tracker on target?

A fundamental yet very challenging task is to determine when tracking is successful: one needs to establish whether a tracker is *correctly* estimating the target state at each time instant or it is estimating the state of another physical process corresponding to another portion of the image. In the former case, the tracker is *on-target*, whereas in the latter the tracker is *on-background*. We differentiate two cases of tracker-on-background, namely, when the tracker is estimating the state of a distractor (an object with similar features to those of the target) and when the tracker is recovering from a failure.

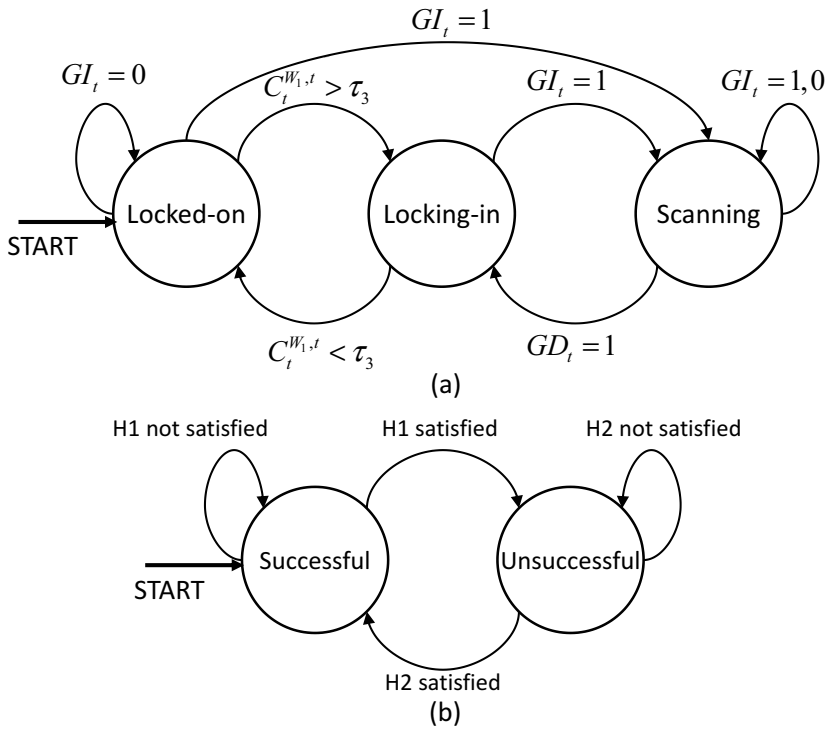


Fig. 8.2. (a) The finite-state machine we use to estimate the tracker condition. The conditions are: *locked-on*, when the algorithm is tracking the target or a distractor (an object similar to the target), *scanning*, when the algorithm is searching the target after a failure, and *locking-in*, when the algorithm is re-focusing on the target or a distractor during a recovery. (b) The finite-state machine used to estimate the temporal segmentation (*successful* and *unsuccessful* tracking).

8.2.1 Problem modeling

To describe the tracker condition, we define three events, here referred to as *locked-on*, *locking-in* and *scanning*. The *locked-on* event describes the tracker while *following* an object, which can be the target or a distractor. The *locking-in* event refers to the tracker adapting its estimation to an object after a failure or when the track is better adjusted to (closing-in) the target. Finally, the *scanning* event describes the tracker searching an object after a tracking failure has happened.

We model the determination of the tracker condition using a Finite-State Machine (FSM). A FSM is represented by a directed graph $\mathcal{G} = \langle \mathcal{S}, \mathcal{E} \rangle$ where $\mathcal{S}$ is the set of nodes defining the states and $\mathcal{E}$ is the set of transitions from one state to another. The state diagram of the tracker-condition FSM is depicted in Fig. 8.2(a).

Next, based on the established tracker condition, we segment time intervals of operation of a tracker based on whether the algorithm is *on-target* (successful) or on *on-background* (unsuccessful). This temporal segmentation is modeled with a second FSM whose state diagram is depicted in Fig. 8.2(b). The transitions between the states of the two FSMs are defined as described in the following sections.

8.2.2 Uncertainty analysis

Let the target state, x_t, at time t be defined as:

$$x_t = f(I_t,\, x_{t-1},\, \beta_{t-1}), \tag{8.1}$$

where $f(.)$ represents the tracking algorithm, I_t the video frame at time t, β_{t-1} the model of the target[2] to track at time $t-1$ and x_{t-1} the target state estimation at time $t-1$.

Based on its widespread use in video tracking, let us consider Bayesian filtering as example of a tracker. In particular we will use a framework defined for elliptical color-based particle filtering [Nummiaro et al., 2003]. The state x_t is a vector whose elements define the position, the two axes and the orientation of an ellipse on the image plane, whereas the model β_{t-1} is a color histogram. The output of the filter at each time step is the sample set $X_t = \left\{(\mathbf{x}_t^{(n)}, \pi_t^{(n)})\right\}_{n=1,\ldots,N}$ of N weighted particles, where each particle $\mathbf{x}_t^{(n)}$ represents one hypothetical state of the target and it is weighted by $\pi_t^{(n)}$, according to the similarity of its features to those of the model.

The uncertainty of the tracking filter can be used as indicator of unstable periods of the output data (e.g., wrong target estimation) providing information about the tracker conditions mentioned in the previous section. We measure the tracker uncertainty using the spatial uncertainty of the N particles. This uncertainty can be estimated by analyzing the eigenvalues of the covariance matrix $C_t = (C_{ij})$ [Maggio et al., 2007], where for simplicity of notation we omit the time index t from each element of the matrix. Each element of the matrix is defined as:

$$C_{ij} = \sum_{n=1}^{N} \pi^{(n)} \mathrm{E}\left[(x_i^{(n)} - \mu_i)(x_j^{(n)} - \mu_j)\right], \tag{8.2}$$

where $\pi^{(n)}$ is the weight of each particle n; $x_i^{(n)}$ ($x_j^{(n)}$) is the *i-th* (*j-th*) element of the *n-th* particle; N is the number of particles; $i,\, j = 1\,, ...,\, d$; and d is the number of dimensions of the state vector. In the specific case mentioned above with the state composed of five elements, we compute a 5x5 covariance matrix. Consequently, the spatial uncertainty, S_t, is defined as [Maggio et al., 2007]:

$$S_t = \sqrt[d]{det(C_t)}, \tag{8.3}$$

where $det(\blacksquare)$ represents the determinant of a matrix.

The tracker uncertainty $\tilde{U}_t$ is finally obtained by normalizing the spatial uncertainty using the width, H_x, and the height, H_y, of the target (i.e. the axes of the ellipse):

$$\tilde{U}_t = \frac{S_t}{4 H_x H_y}. \tag{8.4}$$

[2] In case the model of the target does not change after initialization, then β_{t-1} is replaced by β_{t_0}

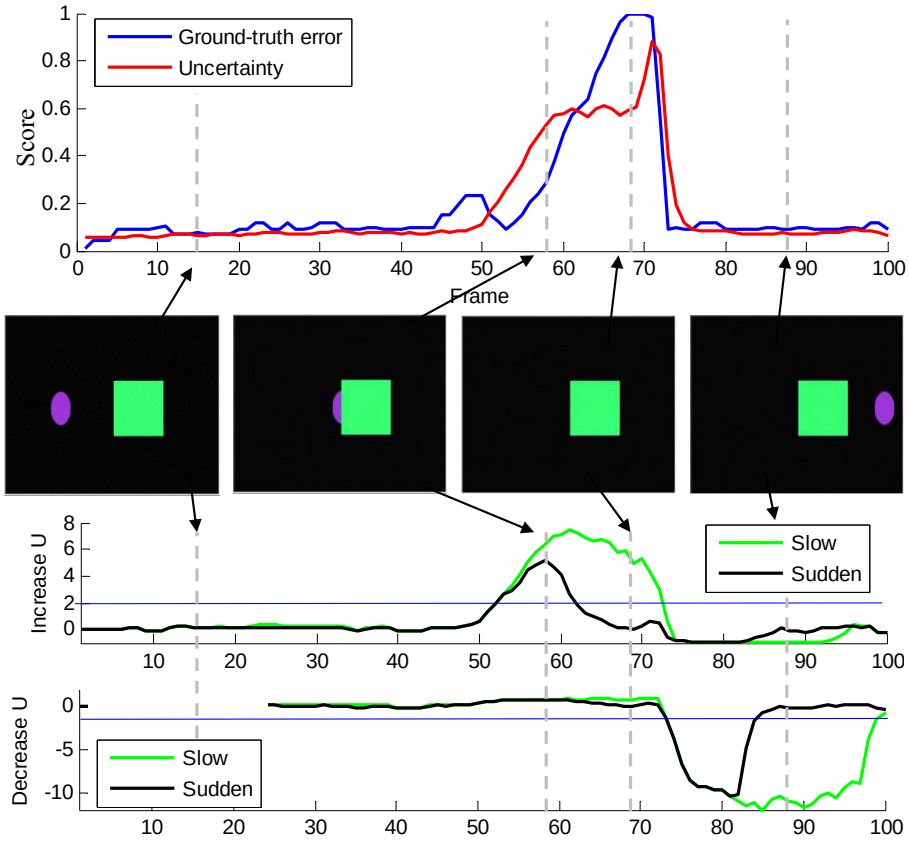


Fig. 8.3. Evolution of tracking filter uncertainty and ground-truth error for a toy sequence. Blue lines in the bottom plots indicate the value of the threshold applied ($\tau_1 = 2$ for increases and $\tau_2 = -2$ for decreases). The ground-truth error signal was computed as described in Sec. 8.4.2.

Note that this uncertainty measure is independent of the target size and of the number of samples. A temporal filtering stage is finally applied to smooth the result:

$$U_t = \alpha U_{t-1} + (1 - \alpha)\tilde{U}_t, \tag{8.5}$$

where $\alpha \in [0, 1]$ determines its update rate. Lower values of α produce a faster update.

The tracker is expected to maintain a constant or slightly decreasing value of uncertainty (indicating a temporal refinement of target estimation) when it is successfully tracking the target over time. The filter uncertainty increases when the tracker loses the target. Finally, a decrease of the filter uncertainty after a tracking failure indicates that the tracker has locked on an object, which might be the correct target or a distractor.

Fig. 8.3 illustrates an example of this uncertainty analysis. The color-based particle filter is applied to track a solid ellipse that moves from left to right and gets occluded for few frames by a square (Fig. 8.3, middle). It can be observed that the interval during which the uncertainty increases and decreases reflects what one could estimate using GT data (Fig. 8.3, top).

8.2.3 Tracker condition estimation

We aim to detect temporal changes in uncertainty levels to discern transitions of the tracker condition between locked and not locked, at each time instant. In fact, small uncertainty levels indicate that the tracker is locked on an object (the target or a distractor), whereas large uncertainty levels imply that the tracker is scanning the image while searching for a suitable candidate to lock on. Moreover, we aim to detect the tracker adaptation to wrong objects (distractors).

To detect uncertainty level transitions while removing the offset value that could be exhibited by the tracking algorithm, we define a change signal, C_t^W, that maximizes the difference between the filter uncertainty at time t, U_t, and previous uncertainty values within a time window W:

$$C_t^{W,k} = \frac{U_t - U_{\hat{t}}}{U_k}, \tag{8.6}$$

where

$$\hat{t} = \underset{j \in W}{argmax} \left(\frac{U_t - U_j}{U_k} \right) \tag{8.7}$$

and $k \in \{\hat{t}, t\}$, with $k = \hat{t}$ for detecting low-to-high uncertainty level transitions and $k = t$ for detecting high-to-low uncertainty level transitions. The size of the time window determines the speed of response of the operator. Large windows allow to detect slow changes but they introduce a delay in the filter response when the signal recovers the non-change condition. Small windows allow to detect sudden changes providing a quick operator response but they are sensitive to the signal rate change and therefore slow-changing signals are not detected by using small windows.

Slow and sudden changes in the signal are detected by using two different temporal window sizes, W_1 and W_2. These combinations generate four change signals: $C_t^{W_1,\hat{t}}$, $C_t^{W_2,\hat{t}}$, $C_t^{W_1,t}$, and $C_t^{W_2,t}$ that monitor slow (W_1) or sudden (W_2) increases ($k = \hat{t}$) or decreases ($k = t$) of the uncertainty. Examples of these signals are shown in Fig. 8.3, bottom.

We define transitions between tracker conditions based on changes of the uncertainty signals to detect global and local changes. The global change conditions, GI_t and GD_t, are defined as:

$$GI_t = \begin{cases} 1 & if \quad C_t^{W_1,\hat{t}} \geq \tau_1 \quad \vee \quad C_t^{W_2,\hat{t}} \geq \tau_1 \\ 0 & otherwise \end{cases} \tag{8.8}$$

and

$$GD_t = \begin{cases} 1 & if \quad C_t^{W_1,t} \geq \tau_2 \quad \vee \quad C_t^{W_2,t} \geq \tau_2 \\ 0 & otherwise \end{cases} \tag{8.9}$$

where τ_i $(i = 1, 2)$ represent relative changes (e.g. $\tau_i = 2$ indicates a 200% change).

The proposed tracker-condition FSM model (Fig. 8.2(a)) starts in the *locked-on* state when the tracker is initialized. Then, it jumps to the *scanning* state when a global increase is detected,

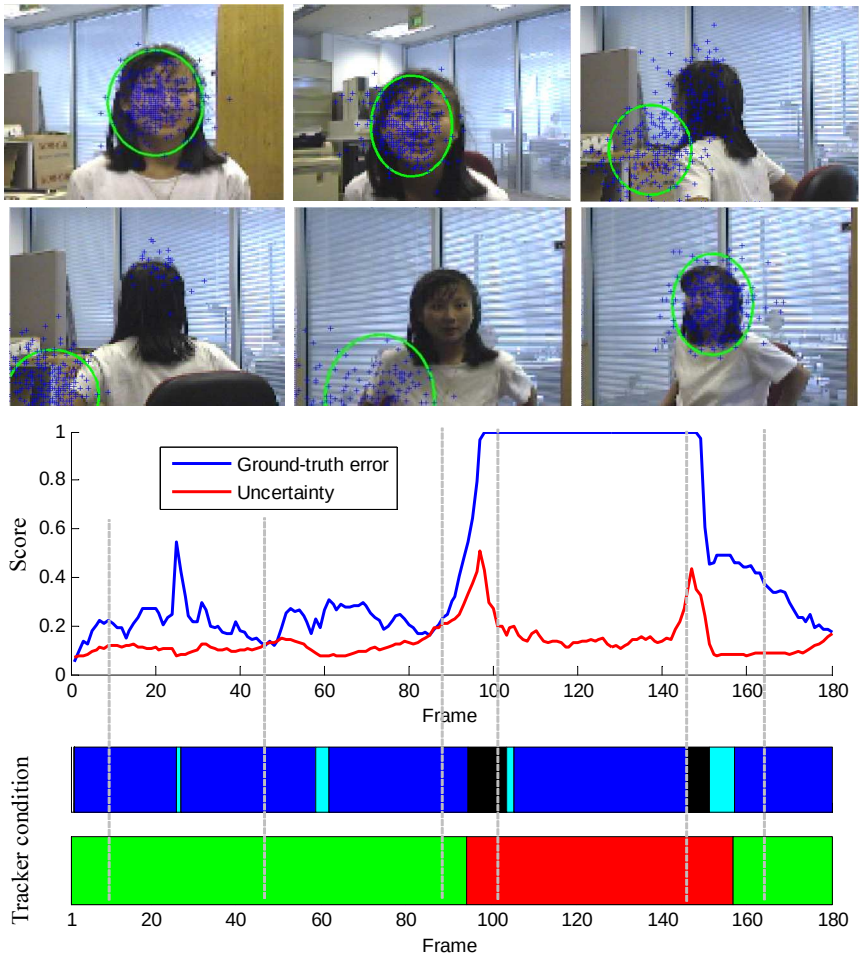


Fig. 8.4. Sample tracking results, filter uncertainty, ground-truth error, tracker condition estimation and temporal segmentation for the test sequence *seq_ mb* (frames 10, 45, 88, 103, 147 and 165). Tracking results and the spatial localization of particles are represented as, respectively, green ellipses and blue crosses. The ground-truth error signal was computed as described in section 8.4.2. (Green: successful tracking; Red: unsuccessful tracking; Black: scanning; Cyan: locking in; Blue: locked on.)

$GI_t = 1$, and to the *locking-in* state when a (small) sudden uncertainty decrease is detected, $C_t^{W_2,t} > \tau_3$. τ_3 evaluates the amount of decrease change (e.g., $\tau_3 = \tau_2/2$). In the *scanning* state, the FSM jumps to the *locking-in* state when a global decrease is detected, $GD_t = 1$. Then, the FSM maintains its state if there is a sudden uncertainty decrease, $C_t^{W_2,t} > \tau_3$, jumps to the *locked-on* state in case of the stabilization of the uncertainty signal, $C_t^{W_2,t} < \tau_3$, or goes to the *scanning* state if a global uncertainty increase is detected, $GI_t = 1$.

Fig. 8.4 shows an example of temporal segmentation of the tracker condition. The FSM determines the behavior of the tracker when the algorithm follows the target and a wrong object (*locked-on* condition), searching for potential candidates after a tracking failure (*scanning* condition) and focusing on the selected target (*locking-in* condition).

8.2.4 Detection of recovery from an error

The analysis of the filter uncertainty alone can not determine a recovery of the tracker after a tracking failure. In fact, locking on a wrong object (distractor) may occur because of similarities between the target model and the features of other objects in the scene. In this situation, the uncertainty level of the tracker correctly following the target might be the same as the level of the tracker following a distractor (see Fig. 8.4).

To overcome this limitation and to provide an accurate detection of the recovery after a tracking failure, we propose to use the time-reversibility property [Wu et al., 2010]. Time reversibility assumes that the movement performed by an object over time (forward) is also exhibited in the reverse direction and the tracker shall be able to track the target in the reverse (backward) direction. In order to describe the tracking process in the forward and the reverse direction, let

$$x_t^F = f^F(I_t, x_{t-1}^F, \beta_{t-1}^F) \tag{8.10}$$

be the state estimated using a forward tracking process and

$$x_t^R = f^R(I_t, x_{t+1}^R, \beta_{t+1}^R) \tag{8.11}$$

be the state estimated using a reverse tracking process; the superscripts F and R indicate the forward and reverse processes and their related variables; x_t, I_t and β_t are, respectively, the target state, the current frame and the target model at time t; and f is the tracking algorithm.

At each time, the recovery analysis is performed when there is a transition from the condition *scanning* to the condition *locked-on*, through the *locking-in* condition. In this case, the time-reversed tracking analysis is started and initialized with the current target state estimate (obtained with the forward tracker). A reference point is defined for the reverse analysis (determining its length) as the last known time of the forward tracker estimation when the target state was correctly estimated (successful tracking) before the target was lost (unsuccessful tracking). Therefore, the previously determined values of successful/unsuccessful forward tracking are saved to choose the appropriate reference point. As the forward estimation at the reference point usually contains little information about the target, the real reference point is selected as the furthermost point in the previous half a second that was determined as successful tracking.

Then, the forward and reverse target estimations are compared to detect the recovery after failure. To measure the overlap between two spatial locations (extracted from the target estimations), we use the dice coefficient [Nghiem et al., 2007], which is defined as follows:

$$d_S(x_{t_0}^F, x_t^R) = \frac{2\left|A_t^F \cap A_t^R\right|}{\left|A_{t_0}^F\right| + \left|A_{t_0}^R\right|}, \tag{8.12}$$

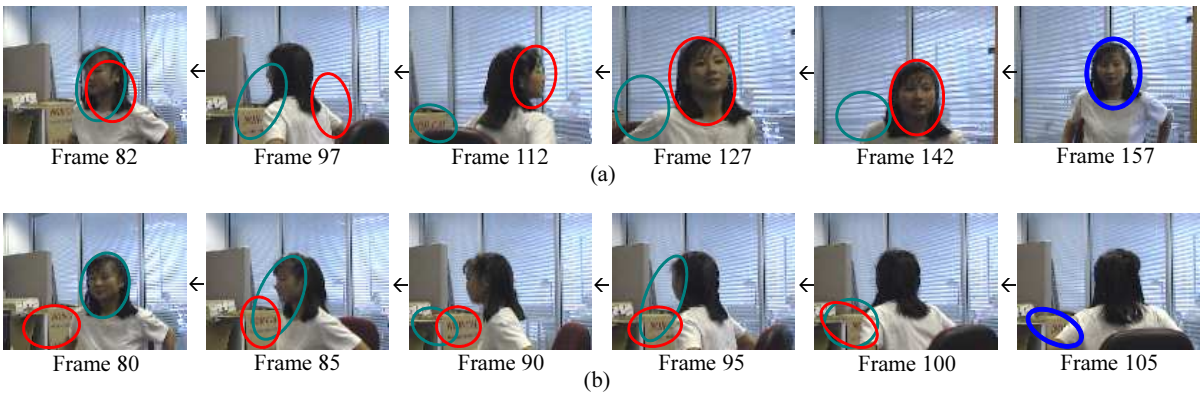


Fig. 8.5. Examples of reverse tracking applied to detect the recovery from error for the test sequence *seq_mb*. (a) Target recovery after failure in frame 157. (b) Wrong adaptation to a distractor in frame 105. (Green ellipse: estimation using forward tracking; Red ellipse: estimation using reverse tracking; Blue ellipse: evaluation of track recovery.)

where t_0 is the reference point to check the similarities between the forward and the reverse tracking estimates; $x_{t_0}^F$ and $x_{t_0}^R$ are the reverse and forward target state estimations at time t; $|A_t^F \cap A_t^R|$ is their spatial overlap in pixels; and $|A_t^F|$ and $|A_t^R|$ are their area in pixels. We detect a tracker recovery if the value of $d_S(x_{t_0}^F, x_t^R)$ is above a certain threshold, τ_4.

Fig. 8.5 shows two examples for detecting a tracking recovery. As previously observed in Fig. 8.4, the uncertainty analysis determined that the tracker became unstable around frames 95-100 and 140-150. Few frames later, the uncertainty stabilized in both cases (Fig. 8.5(a)) recovering from error and (Fig. 8.5(b)) adapting to a wrong target. In both situations, the proposed method was able to detect the (a) correct and (b) wrong recoveries after error.

8.2.5 Tracker operation condition

Finally, the operation conditions during which the tracker is performing successfully or unsuccessfully are defined based on transitions dependent on two conditions, H_1 and H_2 (Fig. 8.2(b)). Let us assume that the tracker starts from a successful state when it is initialized. Then H_1 is satisfied when the tracker condition moves to or remains in *scanning*. H_2 is satisfied when the tracker condition moves from *locking-in* to *locked-on* and there is a correct recovery from error (i.e., $d_S(x_{t_0}^F, x_t^R) \geq \tau_4$).

An example of temporal segmentation defining the operation condition of a tracker is shown in Fig. 8.4, bottom. As there are similar objects in the background and the target changes its appearance, the tracker is not able to perform successfully and a failure happens between frames 90 and 160. The temporal segmentation of successful tracking is correctly performed by combining the tracker condition results and the accurate detection from recovery.

8.3 Track-quality estimation

After the temporal segmentation of successful and unsuccessful tracking, track quality is estimated for the temporal segments during which the tracker is successful. The others segments are considered track-lost segments and therefore discarded for measuring the accuracy of the tracker [Maggio and Cavallaro, 2011]. We apply the time-reversibility constraint [Wu et al., 2010] and measure at each time step the similarities between the state estimated with the forward tracker and the state estimated with the reverse tracker (see Eqs. 8.10 and 8.11).

For each evaluation time, a reverse tracker is created and initialized with the current target estimation obtained from the forward tracker (the tracker to be evaluated). Then, tracking is performed in reverse direction until a reference frame defined as the frame where the last successful recovery from error was detected (see subsection 8.2.4). The initial frame of the video is considered as the first reference frame. Note that the forward and reverse trackers have to be defined using the same tracking algorithm in order to maintain the time-reversibility property.

Then, the differences between the forward and reverse tracker are used to estimate the quality, Q_t, of the current track as:

$$Q_t = 1 - \frac{1}{t - t_1} \sum_{i=t_1}^{t} D(x_i^F, x_i^R), \tag{8.13}$$

where x_t^F and x_t^R are the target state estimations from, respectively, the forward and reverse tracking; t_1 is the reference frame for the reverse tracking analysis; and $D(.)$ is the function that measures the dissimilarity between the forward and the reverse analysis. Inspired by the improvement achieved by the hybrid evaluation approaches (e.g., [Kalal et al., 2010; Pan et al., 2009a]), we use a combination of features to generate the dissimilarity measure, $D(.)$:

$$D(x_t^F, x_t^R) = \omega d_S(x_t^F, x_t^R) + (1 - \omega) d_F(x_t^F, x_t^R), \tag{8.14}$$

where $\omega \in [0, 1]$ is the weight that combines the distances; $d_S(x_t^F, x_t^R)$ is defined in Eq. 8.12; and $d_F(x_t^F, x_t^R)$ is a *feature* distance that for the elliptic color tracker we define as

$$d_F(x_t^F, x_t^R) = \sqrt{1 - \rho(\mathbf{p}, \mathbf{q})}, \tag{8.15}$$

where

$$\rho(\mathbf{p}, \mathbf{q}) = \sum_{u=1}^{m} \sqrt{p_u q_u} \tag{8.16}$$

is the Bhattacharyya coefficient computed between the m–bin color histograms $\mathbf{p}$ and $\mathbf{q}$ of the forward and reverse target estimations. A high value of ω should be selected where there is a noticeable clutter level because the color histograms of the target will not provide an accurate color representation. Therefore, increasing the weight of the *spatial* distance will increase the

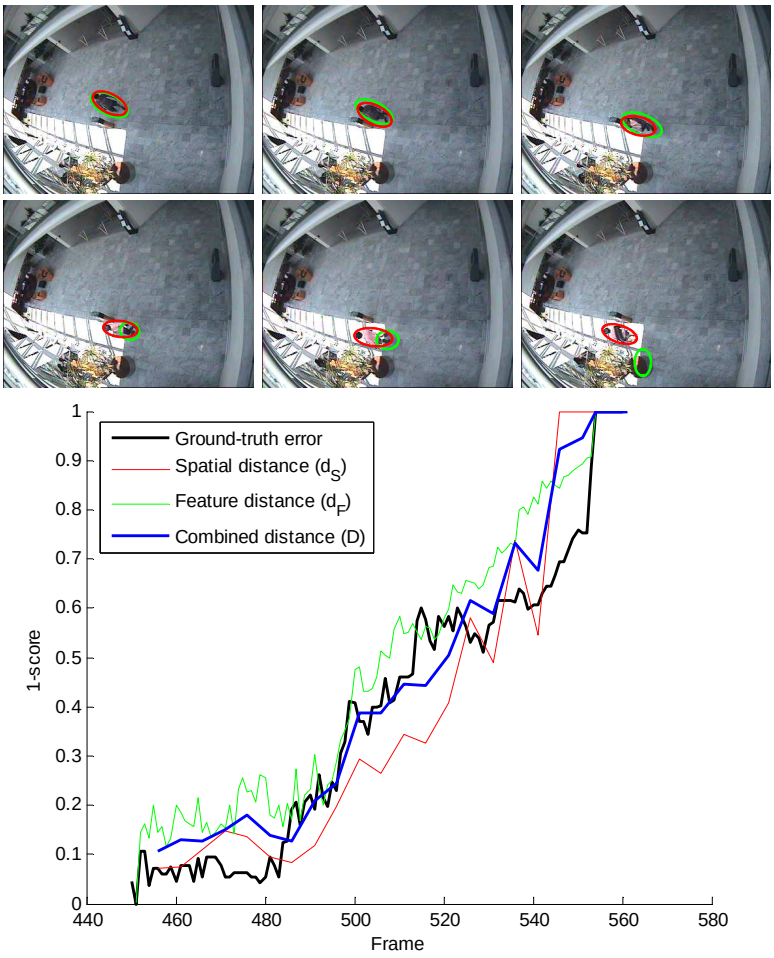


Fig. 8.6. Comparison of proposed distances to measure track quality. Sample images shown correspond to frames 460, 480, 500, 520, 540 and 560. Tracking results and ground-truth annotation are represented as green and red ellipses, respectively. The ground-truth error is measured as the spatial overlap between estimated and ground-truth target (Eq. 8.12).

performance of the estimated track quality. Although the clutter analysis is a difficult task, it can be approximated using the discriminative feature power proposed in [Collins et al., 2005]. On the other hand, lower values of ω will increase the weight of the *feature* distance that could be useful if the tracker is not able to accurately determine target positions in forward and reverse directions. For a generic scenario, we assume that both distances have an equal impact on the track quality, hence $\omega = 0.5$.

An example of track quality estimation is shown in Fig. 8.6. The progressive decrease in performance (measured with the GT error) is well approximated by the proposed distances. The forward tracking result (depicted as green ellipses) was used to initialize the reverse tracking and compute the similarity scores. The reverse analysis was performed until the initialization frame of the target used in the example (frame 450).

Dataset	Target	Size	Characteristics
CAVIAR	P1 – P4	384x288	IC, C
PETS2001	P5 – P10	768x576	SC, O, C
PETS2010	P12 – P18	768x576	O, C
CLEMSON	F1 – F4	128x196	SC, AC, C, O
VISOR	F5, F6	352x288	SC, C, O

Table 8.1: Description of the evaluation dataset (SC: Scale changes. AC: Appearance changes. IC: Illumination changes. O: Occlusions. C: Clutter.)

8.4 Experimental results

8.4.1 Experimental setup

We evaluate the results of the proposed approach, ARTE (Adaptive Reverse Tracking Evaluation)[3], and compare it with representative state-of-the-art approaches for empirical standalone quality evaluation: observation likelihood (OL) [Vaswani, 2007], covariance of the target state (SU) [Maggio et al., 2007], frame-by-frame reverse-tracking evaluation using template inverse matching (TIM) [Liu et al., 2008] and full-length reverse-tracking evaluation using the same applied tracking algorithm (FBF) [Wu et al., 2010]. Compared to the measures selected in the previous chapter (see section 7.4), we have excluded the Motion Smoothness measure (MS) due to the low performance observed in the experiments, we have replaced the COV measure by the SU one (for avoiding the covariance normalization carried out by the COV measure) and we have included the FBF measure that extends the TIM approach using a full-length reverse tracker.

The evaluation dataset is composed of sequences from CAVIAR[4], PETS2001[5], PETS2010[6], the CLEMSON dataset for face tracking[7] and VISOR[8]. These sequences present challenging situations for tracking such as total or partial occlusions, clutter and illumination or scale changes (Table 8.1). The initialization of each target is shown in Fig. 8.7. To evaluate the performance, we use GT information consisting on the ellipse that best fits the target at each frame and it is described by its centroid coordinates, size of the axes and rotation angle.

The tracker parameters [Nummiaro et al., 2003] were the same for all the targets (heuristically set to $\sigma_{x,y} = 5$, $\sigma_{H_x,H_y} = 0.75$, $\sigma_\theta = 4°$, $\sigma_c = 0.2$ and $N = 300$). Color histograms were generated in the RGB and HSV spaces for, respectively, pedestrian and face targets using 8x8x8 bins in both cases. The change detection thresholds were empirically set to $\tau_1 = 2$ and $\tau_2 = -2.5$. Due to the statistical nature of the filter, we run the tracker 10 times for each sequence.

[3] Additional results and test data can be found at http://www-vpu.eps.uam.es/publications/TrackQuality
[4] http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/
[5] http://www.cvg.rdg.ac.uk/PETS2001/
[6] http://www.cvg.rdg.ac.uk/PETS2010/
[7] http://www.ces.clemson.edu/~stb/research/facetracker
[8] http://imagelab.ing.unimore.it/visor/

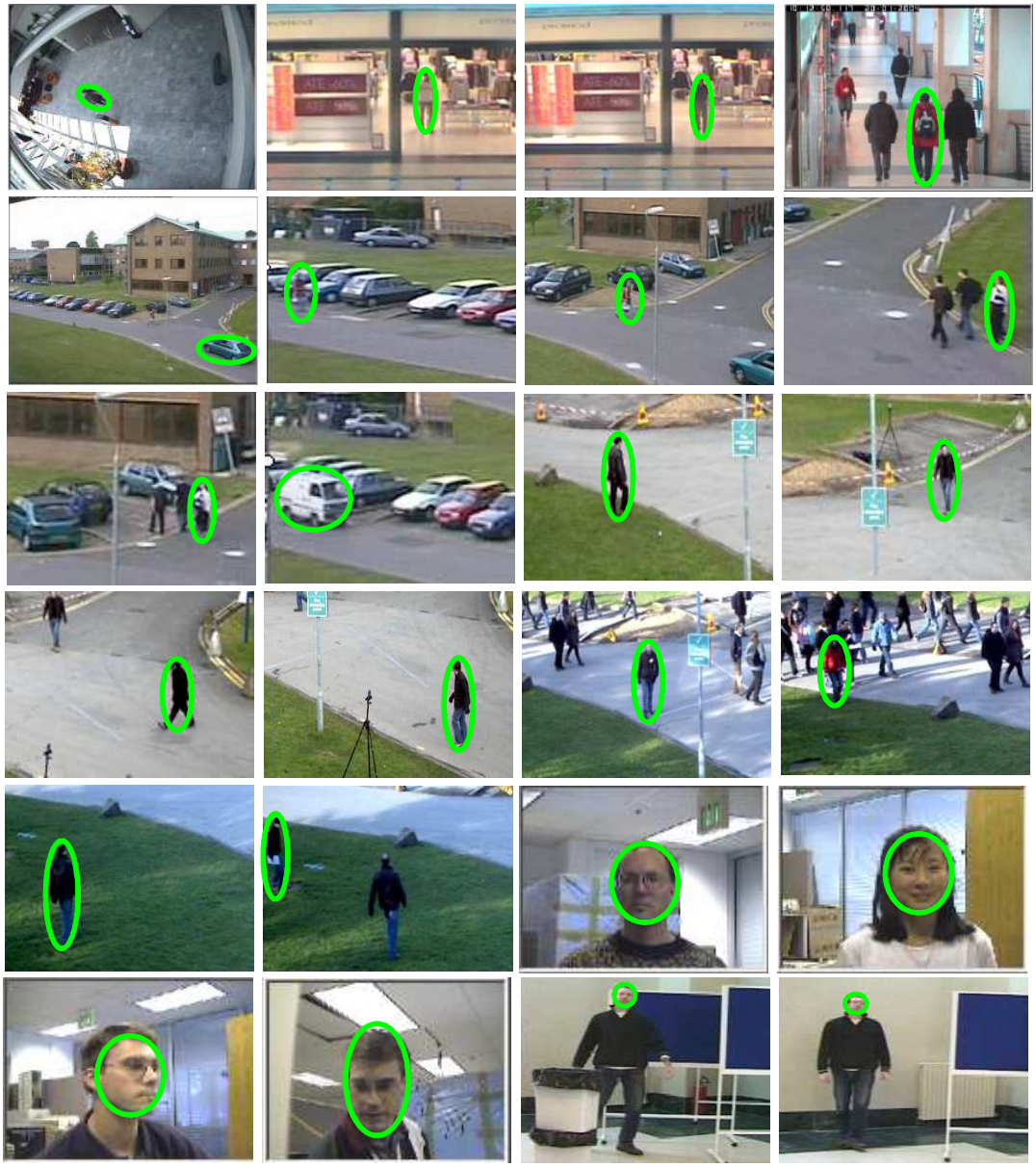


Fig. 8.7. Target initialization for the evaluation dataset. From top-left to bottom-right: Pedestrian targets: *Browse_WhileWaiting1* (P1), *OneLeaveShopReenter1front* (P2), *OneLeaveShopReenter2front* (P3), *ThreePastShop2cor* (P4), *Camera1_testing* (P5–P10), *S2_L1_view001* (P11–P14), *S2_L2_view0001* (P15, P16) and *S2_L3_view001* (P17, P18); Face targets: *seq_bb* (F1), *seq_mb* (F2), *seq_sb* (F3), *seq_villains2* (F4), *occlusion_1* (F5) and *occlusion_2* (F6).

Approach	AUC	FPR	TPR
OL	$.66 \pm .07$	$.37 \pm .05$	$.61 \pm .11$
SU	$.76 \pm .04$	$.38 \pm .03$	$.81 \pm .06$
TIM	$.44 \pm .01$	$.28 \pm .02$	$.27 \pm .04$
FBF	$.87 \pm .03$	$.25 \pm .03$	$.95 \pm .02$
ARTE	$.87 \pm .02$	$.16 \pm .01$	$.89 \pm .03$

Table 8.2: ROC analysis for the temporal segmentation into successful and unsuccessful tracking using 10 runs expressed as mean ± standard deviation. (AUC: area under the curve; FPR: false positive rate; TPR: true positive rate; OL: Observation Likelihood [Vaswani, 2007]; SU: Covariance of the target state [Maggio et al., 2007]; TIM: frame-by-frame reverse tracking [Liu et al., 2008]; FBF: full reverse tracking [Wu et al., 2010]; ARTE: proposed approach.)

8.4.2 Performance evaluation criteria

The error between the tracking data and the GT data is quantified using the spatial overlap of the corresponding target areas (Eq. 8.12). Low performance is indicated by values close to 1 (i.e. small overlap). High performance is indicated by values close to 0 (i.e. large overlap). Track quality is evaluated once every five frames.

The performance of the temporal segmentation (see section 8.2) is evaluated using ROC analysis. An unsuccessful track is determined when the error measure $d_S(x_t^e, x_t^g)$ (defined as in Eq. 8.12) is larger than the minimum allowed overlap between, x_t^g, the GT area, and x_t^e, the detected area (e.g. 0.5 indicates an overlap of 50%).

Finally, the performance of the track quality estimation is evaluated by the correlation with the GT error for the case of successful tracking (determined using ARTE). We use the Pearson product-moment correlation coefficient [Chen and Popovich, 2002] as described in Eq. 7.16.

8.4.3 Temporal segmentation to detect successful tracking

The results of the temporal segmentation between successful and unsuccessful tracking are summarized in Fig. 8.8 and Table 8.2. Feature-based measures (OL and SU) demonstrated their dependency with the level of clutter (for OL) and with the adaptation to wrong targets (OL and SU), thus obtaining intermediate results. SU obtained better results as it relies on the filter uncertainty. TIM showed the inaccuracy of short-length reverse-based evaluation due to the adaptation of the measure to the tracking errors. On the other hand, the time-reversibility property is useful to segment correct tracking and in fact the full-length reverse-based evaluation (FBF) obtained high performance. ARTE had similar AUC compared to FBF. An intersection of both ROC curves shows that the FBF outperforms the ARTE approach with higher *true positive rate*. However, the observed *false alarm rate* of FBF is also higher than for ARTE. On the other hand, ARTE obtained better *true positive rate* than FBF in the case of low *false alarm rate*.

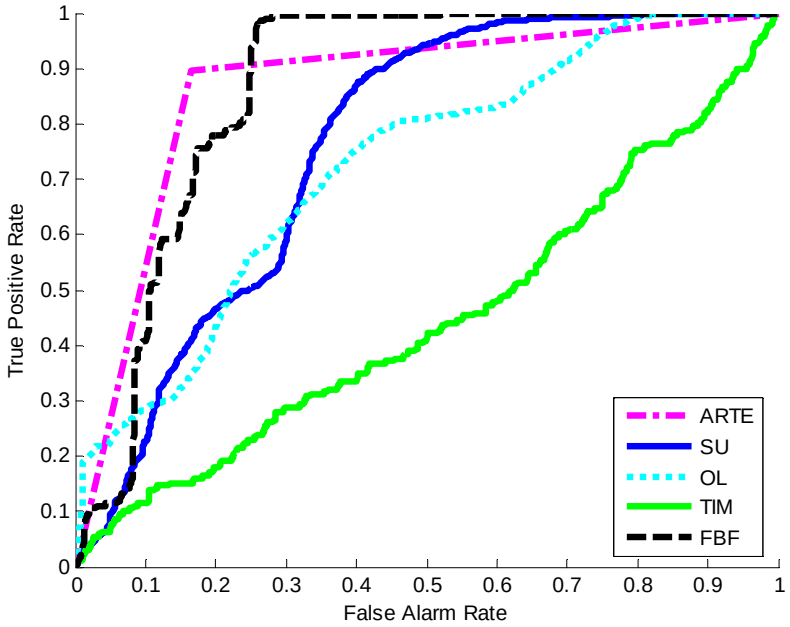


Fig. 8.8. ROC curves for the segmentation between successful and unsuccessful tracking using the evaluation dataset. (OL: Observation Likelihood [Vaswani, 2007]; SU: Covariance of the target state [Maggio et al., 2007]; TIM: frame-by-frame reverse tracking [Liu et al., 2008]; FBF: full reverse tracking [Wu et al., 2010]; ARTE: proposed approach.)

Computational cost results of the temporal segmentation are listed in Table 8.3. Two groups can be differentiated attending to the similarity between the *min*, *max* and *mean* values. The first group (OL, SU and TIM) presented a quasi-constant cost per frame as they always did the same operations. OL and SU obtained low values due to their low complex operations. High TIM values are explained by the template matching process of the tracking analysis from the current to the previous frame. On the other hand, the second group (FBF and ARTE) showed a variable cost. For every frame, FBF employs a reverse analysis that is performed until a reference point (fixed to the initialization frame of the target) is reached. As the length of the sequence increases, the reverse analysis cost also grows. Hence, the cost of FBF has an exponential increase rate that depends on sequence length making it inadequate for tracking evaluation in long sequences. In particular, the *min* (*max*) FBF value was obtained in the first (last) frame of the sequence. For ARTE, the uncertainty analysis had a constant cost whereas the error recovery presented a complexity-dependent cost. In complex sequences, the tracker is expected to have more failures and recoveries requiring to check the correct recovery. However, the checking process is done once per recovery and the reference points of the reverse analysis are adaptively determined. In conclusion, the computational cost of our approach is considerable lower than the FBF approach.

Fig. 8.9 depicts a case of failure and wrong target adaptation: the tracker starts in the *locked-on* condition and then it moves to an over-illuminated area. Here the tracker loses the

Approach	Execution time per frame (ms)		
	Min	Max	Mean
OL	3.4	3.9	3.7
SU	4.6	5.5	5.2
TIM	13.5	18.5	16.2
FBF	15.6	2502.1	250.2
ARTE	6.2	1050.5	25.2

Table 8.3: Comparative execution time of the temporal segmentation using 10 runs. (OL: Observation Likelihood [Vaswani, 2007]; SU: Covariance of the target state [Maggio et al., 2007]; TIM: frame-by-frame reverse tracking [Liu et al., 2008]; FBF: full reverse tracking [Wu et al., 2010]; ARTE: proposed approach)

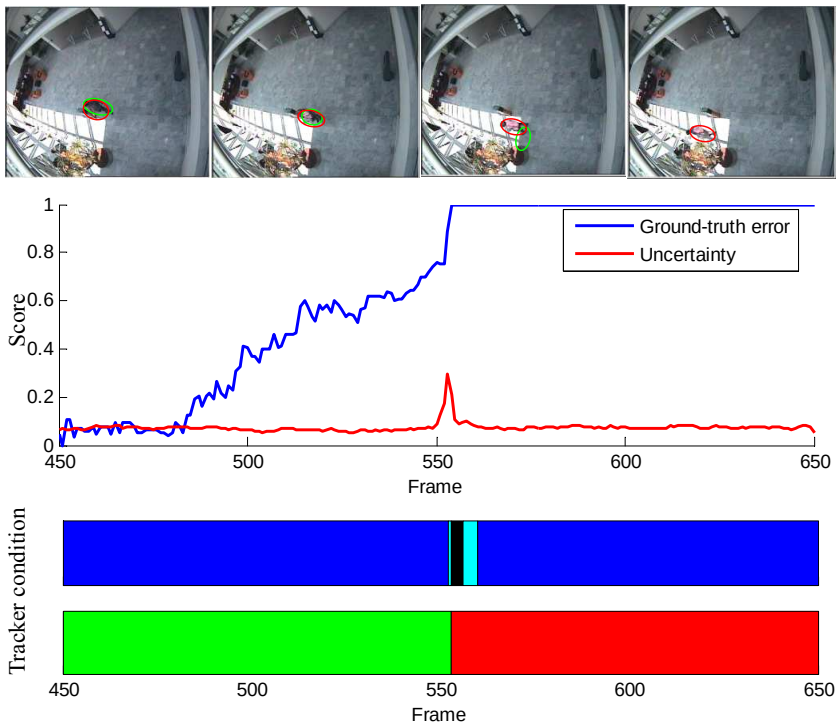


Fig. 8.9. Tracking results, tracker condition estimation and temporal segmentation for target P1 (*Browse_ While Waiting1* sequence; frames shown are 525, 540, 560 and 580). Tracking results and ground-truth annotations are represented as green and red ellipses, respectively. (Green: successful tracking; Red: unsuccessful tracking; Black: scanning; Cyan: locking in; Blue: locked on.)

target (tracker condition *scanning*). Few frames later, the tracker is distracted by a background object (tracker condition *locking-in*). Finally, the tracker is completely adapted to the wrong object (tracker condition *locked-on*). A reverse tracking analysis is performed to check the correct recovery after error and fails indicating the wrong target adaptation.

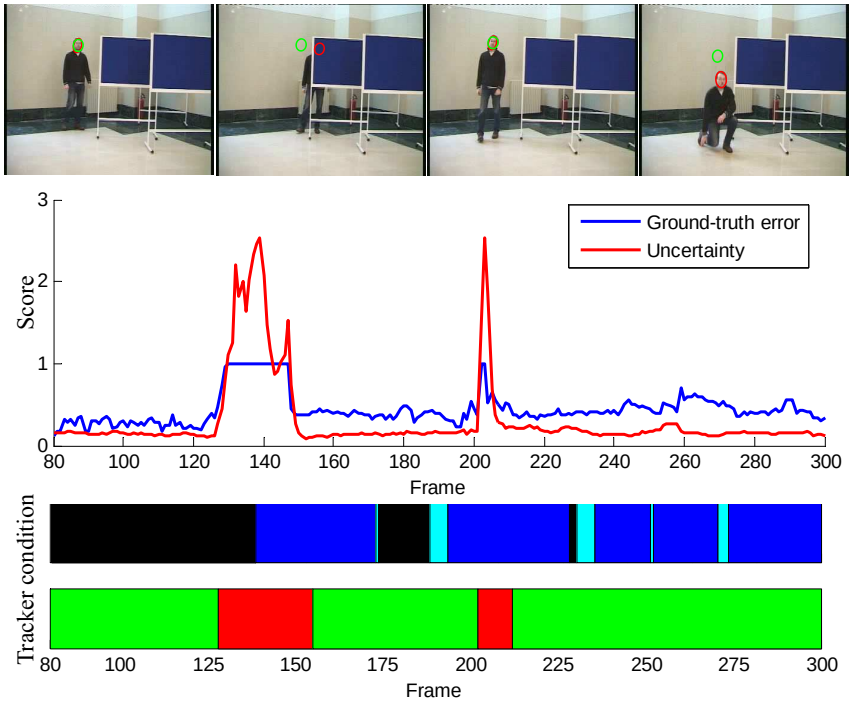


Fig. 8.10. Tracking results, tracker condition estimation and temporal segmentation for target H5 (*occlusion_1* sequence; frames shown are 100, 140, 180 and 210). Tracking results and ground-truth annotations are represented as green and red ellipses, respectively. (Green: successful tracking; Red: unsuccessful tracking; Black: scanning; Cyan: locking in; Blue: locked on.)

Fig. 8.10 shows another temporal segmentation example with failure recovery. A moving head is tracked (tracker condition *locked-on*) until it gets occluded by a blackboard (tracker condition *scanning*). Then, the tracker recovers the correct target (tracker conditions *locking-in* and *locked-on*). Successful recovery is verified by the reverse-based proposed method. Then, the target is again lost due to a quick movement (tracker condition *scanning*) and recovered few frames later (tracker conditions *locking-in* and *locked-on*). Finally, a target estimation is refined as the filter uncertainty is decreased (the transitions between the *locking-in* and *locked-on* conditions).

8.4.4 Track-quality estimation

Experimental results comparing the correlation between the track quality estimators and GT data are summarized for pedestrian and face targets in Table 8.4. As for pedestrian results, ARTE achieved an average GT correlation of 57.3% whilst the other selected approaches obtained correlations of 50.5% (OL), 48.0% (SU), 11.3% (TIM) and 33.15% (FBF). OL obtained diverse correlation values showing its dependency to wrong target adaptation and the different level of clutter. The first situation can be easily observed for P1, P13 and P18, whilst the second situation is observed in P4 and P5 (where there is no tracking failure). In particular, high performance was achieved in case of correct tracking or failures due to target model dissimilarities. OL

Target	Pearson correlation coefficient				
	OL	SU	TIM	FBF	ARTE
P1	**.86 ± .04**	.25 ± .06	.07 ± .05	.09 ± .05	.75 ± .15
P2	.34 ± .10	.20 ± .09	.10 ± .16	.15 ± .07	**.45 ± .20**
P3	.20 ± .06	.49 ± .08	.10 ± .04	.02 ± .12	**.83 ± .11**
P4	**.64 ± .05**	.53 ± .06	.10 ± .01	.33 ± .17	.46 ± .06
P5	.95 ± .02	**.96 ± .02**	.08 ± .04	.95 ± .03	.86 ± .10
P6	.44 ± .10	**.56 ± .08**	.05 ± .03	.45 ± .20	.47 ± .19
P7	.24 ± .15	.25 ± .11	.05 ± .02	.17 ± .09	**.44 ± .14**
P8	.54 ± .13	**.71 ± .16**	.08 ± .07	.47 ± .21	.48 ± .05
P9	**.57 ± .14**	.54 ± .14	.15 ± .11	.10 ± .09	.55 ± .10
P10	.11 ± .03	.23 ± .08	.12 ± .12	.02 ± .01	**.40 ± .10**
P11	.81 ± .15	.64 ± .11	.32 ± .09	.58 ± .11	**.98 ± .01**
P12	.59 ± .11	.28 ± .05	.06 ± .05	.27 ± .09	**.75 ± .08**
P13	.28 ± .10	**.52 ± .14**	.10 ± .13	.14 ± .05	.44 ± .15
P14	.25 ± .05	.66 ± .12	.17 ± .03	.71 ± .13	**.77 ± .04**
P15	**.59 ± .08**	.32 ± .05	.25 ± .12	.43 ± .07	.48 ± .18
P16	.64 ± .11	.45 ± .10	.11 ± .06	.16 ± .03	**.75 ± .05**
P17	.20 ± .05	.18 ± .04	.20 ± .10	.34 ± .09	**.42 ± .08**
P18	.80 ± .14	.82 ± .09	.24 ± .12	.26 ± .12	**.87 ± .15**
mean	.50 ± .10	.48 ± .08	.11 ± .09	.33 ± .10	**.57 ± .03**
F1	.19 ± .06	.60 ± .11	.12 ± .04	.44 ± .11	**.74 ± .17**
F2	.63 ± .15	.38 ± .26	.07 ± .05	.24 ± .09	**.65 ± .10**
F3	.20 ± .17	.57 ± .15	.06 ± .06	**.79 ± .13**	.34 ± .06
F4	.14 ± .09	**.65 ± .04**	.07 ± .03	.25 ± .08	.37 ± .09
F5	**.71 ± .05**	.63 ± .04	.08 ± .04	.42 ± .15	**.71 ± .08**
F6	.31 ± .11	.17 ± .10	.09 ± .05	.05 ± .03	**.32 ± .21**
mean	.36 ± .12	.50 ± .09	.08 ± .05	.37 ± .11	**.52 ± .12**
tot mean	.47 ± .08	.48 ± .08	.09 ± .08	.34 ± .10	**.56 ± .07**

Table 8.4: Comparison of track quality estimation performance for pedestrian (P1-P18) and face targets (F1-F6). Best results are marked in bold. (OL: Observation Likelihood [Vaswani, 2007]; SU: Covariance of the target state [Maggio et al., 2007]; TIM: frame-by-frame reverse tracking [Liu et al., 2008]; FBF: full reverse tracking [Wu et al., 2010]; ARTE: proposed approach)

obtained the best results, thus confirming the conclusions achieved in chapter 7. SU obtained low performance showing a high dependency on the wrong target adaptation (P1, P2, P3, P6, P7 and P9) and being not able to evaluate track quality (P10) as the particle filter tried to keep it constant during the analysis. However, high performance was obtained for the cases of no track quality degradation (P5) or failure without wrong adaptation or recovery (P8). TIM achieved the worst results as the use of short-length reverse analysis accumulates the error over time. Few

frames after a tracking failure, TIM was not able to evaluate tracking failure as the approach adapts to the target estimation in short frame windows. FBF also presented diverse results as the metric applied (Mahalanobis distance) does not work well with different degradations of track quality. This distance measures data similarity considering their means and covariance matrix. However, probabilistic tracking usually provides weighted estimations of target state (e.g., particle filter weights). Therefore, the small changes in the covariance matrix of the target state are not measured by the Mahalanobis distance. Moreover, this distance does not have a fixed range of values that identifies the case of tracking failure without ambiguities. Hence, several Mahalanobis distance values can correspond to a tracking failure and their correlation with ground-information is low. Additionally, the track quality is computed using only the last estimated state of the reverse analysis (performed until the reference frame). Hence, there is an information loss related with all the non computed reverse-forward comparisons. It obtained high performance for P5 and P14; intermediate performance for P4, P6, P8, P11 and P15. On the other hand, low performance results were obtained for P1, P3, P9, P10, P13 and P16. ARTE addresses all these issues and achieved a good trade-of in all the sequences of the dataset.

Regarding the face target results, ARTE achieved an average GT correlation of 52.8% whilst other approaches obtained correlations of 36.4% (OL), 50.3% (SU), 8.7% (TIM) and 37.7% (FBF). A performance decrease is observed as compared with the pedestrian results due to the higher complexity of the face target sequences. It can be observed that OL and SU obtained intermediate results (similar to ARTE) as compared to the pedestrian target results. This performance can be explained due to the target initialization and the type of sequences. The former regards that a face is easier to annotate than a pedestrian and HSV color space offers a robust modeling of face targets. Hence, the obtained target model is more accurate for faces than for pedestrians. The latter regards the type of tracker failures and recoveries. In most of the cases, the tracker error was due to a temporal occlusion with an object completely different of the target (in appearance terms). Then, the posterior tracker recovery was successful in most cases. In this situation, OL and SU increased their performance as they depend on the similarities between the target model and the candidates. TIM and FBF had low performance due to error accumulation for short-length approaches (TIM) or an inappropriate metric used (FBF), as commented earlier.

Sample results of the track quality estimation are illustrated for wrong target adaptation and correct target recovery after a failure in Fig. 8.11. Fig. 8.11(a) shows how the tracker loses the target adapting to the most similar background. Then, it first recovers the target and again loses it at the end of the sequence. Fig. 8.12(b) shows the tracking of a moving head that gets occluded twice by another moving head and by a blackboard. The first occlusion was by a similar target model (not detected by OL) and the second occlusion was due to a model dissimilarity (correctly detected by all approaches). In this case, track quality is correctly estimated by ARTE only.

As a final remark, ARTE detects a recovery few frames later than it actually happened,

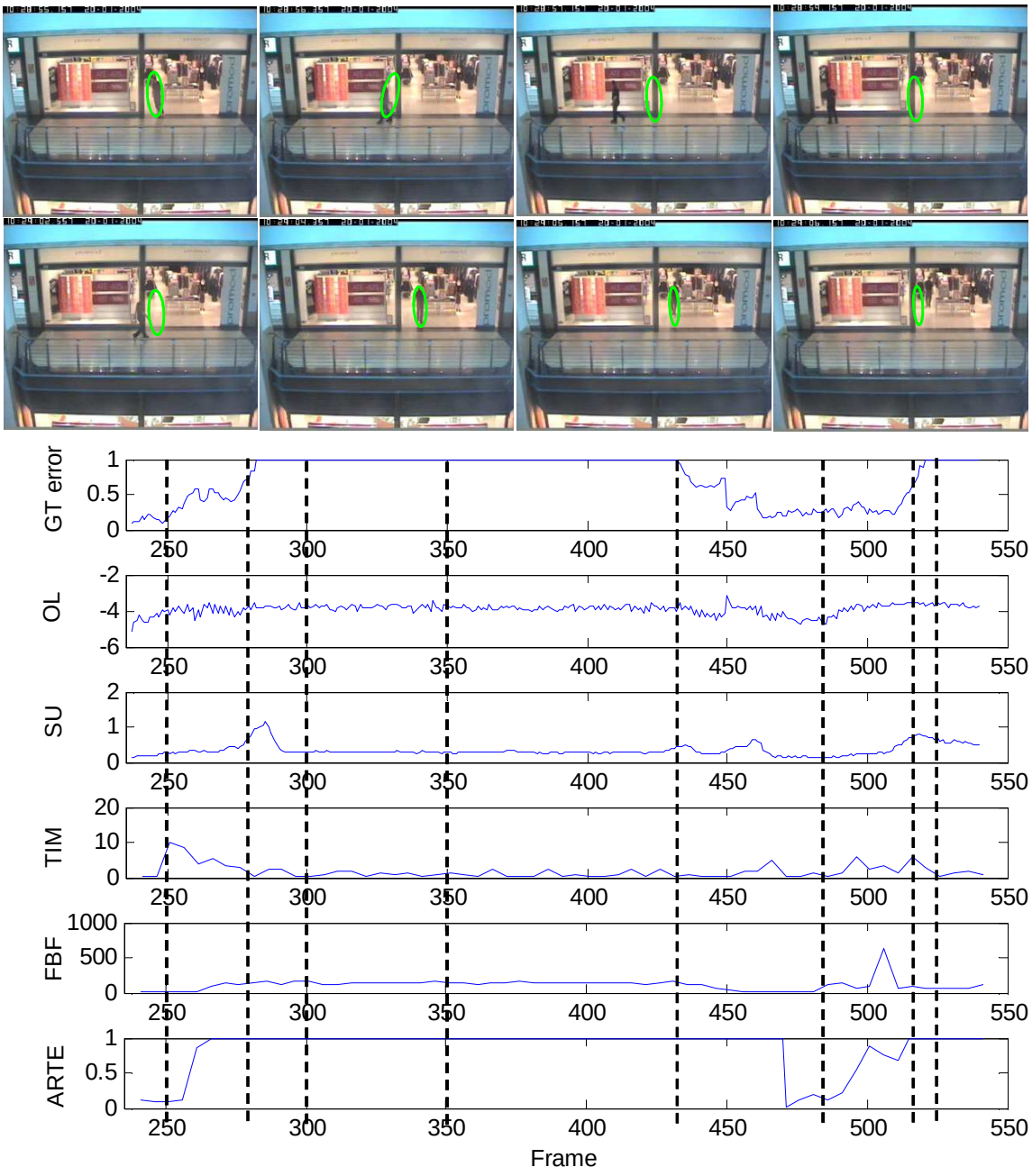


Fig. 8.11. Tracking results, ground-truth error and track quality estimators for target P3 (frames shown are 250, 280, 300, 350, 435, 480, 510 and 525). The methods under analysis are the proposed approach (ARTE), Observation Likelihood (OL) [Vaswani, 2007], Covariance of the target state (SU) [Maggio et al., 2007], frame-by-frame reverse tracking (TIM) [Liu et al., 2008] and full reverse tracking (FBF) [Wu et al., 2010]). Tracking results are represented as a green ellipses.

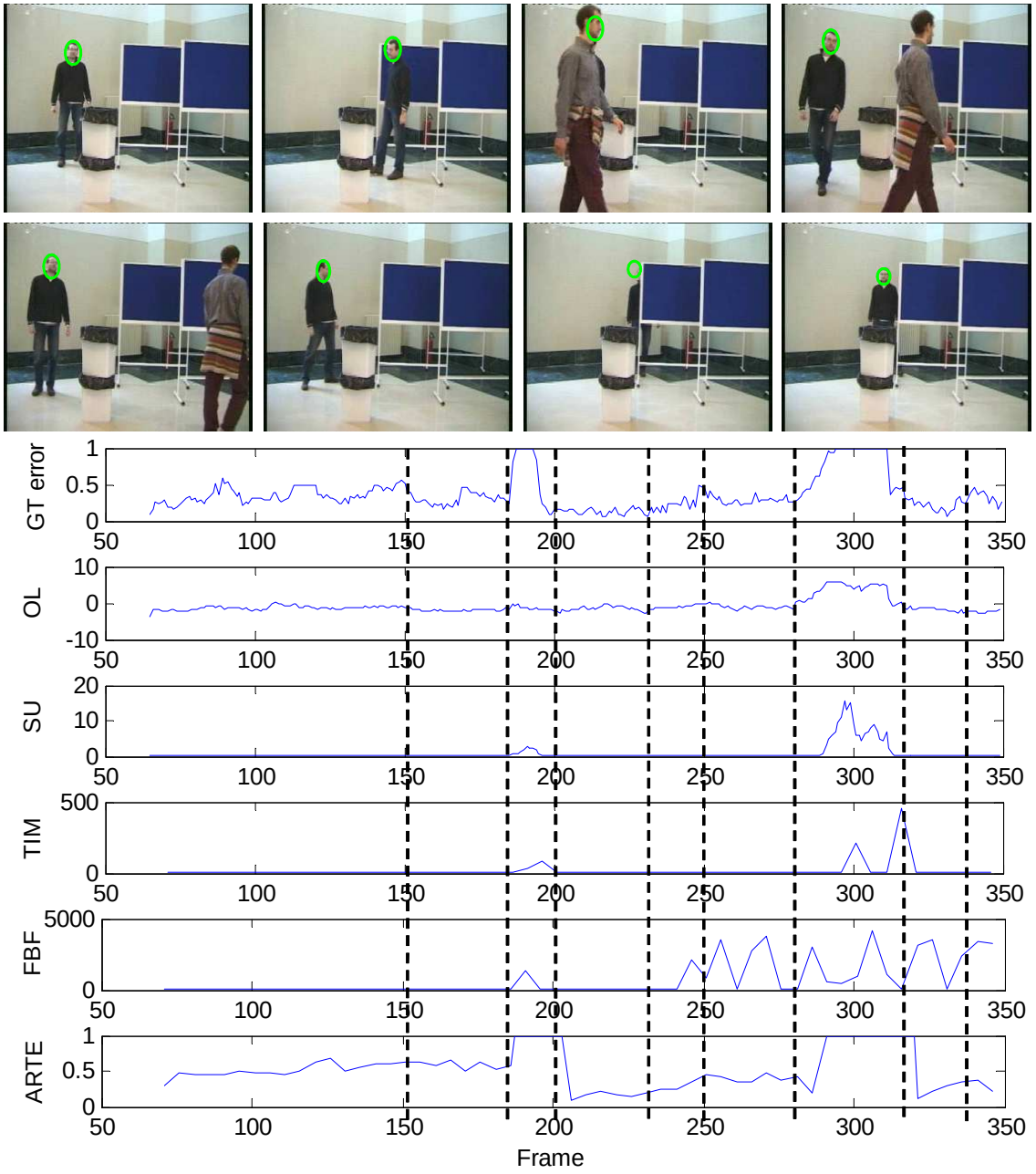


Fig. 8.12. Tracking results, ground-truth error and track quality estimators for target H6 (frames shown are 150, 185, 200, 230, 250, 285, 310 and 340). The methods under analysis are the proposed approach (ARTE), Observation Likelihood (OL) [Vaswani, 2007], Covariance of the target state (SU) [Maggio et al., 2007], frame-by-frame reverse tracking (TIM) [Liu et al., 2008] and full reverse tracking (FBF) [Wu et al., 2010]). Tracking results are represented as a green ellipses.

as shown by the above examples. This latency is due to the filter uncertainty operator that decreases because it integrates previous values. This latency is due to the delayed detection and it can be overcome by changing the size of the applied window: a performance increase (in terms of GT correlation) is expected with a post-processing stage. Nevertheless, a delay in the output of ARTE is introduced to allow the re-calculation of track quality estimations when a recovery is detected. We have decided not to perform this post-processing to avoid an unfair comparison with the state-of-the-art methods as they produce the track-quality data without any post-processing.

8.5 Summary and conclusions

We have introduced a new track quality estimator in the absence of GT information. A novel adaptive analysis strategy based on the uncertainty of the tracking filter and the time-reversibility constraint has been proposed. The idea behind our approach is to identify the status of the target (e.g., target lost, wrong target, looking for the target) and calculate the track quality with a reverse tracker of variable length. Tracking failures are segmented by analyzing the changes of the filter uncertainty. Time-reversibility is applied to check the recovery after a tracking failure. Then, track quality is estimated for the successful tracking cases by using a reverse tracking analysis. It is based on similarities between reverse and forward tracking in terms of color and spatial distances.

Experimental results over a heterogeneous dataset showed that the proposed approach outperforms the state-of-the-art algorithms. It has been demonstrated in the particle filter framework and is valid for any multi-hypothesis tracker that uses some form of uncertainty related to the spread of the generated hypothesis. Results showed that our proposal is able to temporally segment the tracker operation between successful and unsuccessful tracking with high accuracy. Moreover, it presented a low computational cost as compared to the most competing approach (full reverse-based evaluation) that has an exponential grow rate of its cost depending on the sequence length. Hence, our approach is usable for the analysis of long sequences. In addition, our proposal also obtained the best results for track quality estimation. However, the correlation analysis indicated that such estimation presented intermediate accuracy. Therefore, additional features should be employed to provide a more accurate measure of track quality.

Part IV

Conclusions

Chapter 9

Achievements, conclusions and future work

9.1 Summary of achievements and main conclusions

This thesis has addressed the use of the Cognitive Computer Vision paradigm for video analysis. The goal is to increase the robustness of an event recognition system for optimizing its resources and for dealing with unexpected data. In particular, four areas have been explored related with prior knowledge description (chapter 2), structured video analysis (chapters 3 and 4), feedback-based analysis (chapters 5 and 6) and self-evaluation of video analysis (chapters 7 and 8).

Firstly, we have provided a structured representation of the knowledge related to the application domain and the analysis system (chapter 2). Domain knowledge has been described as objects, events and the contextual information whilst system knowledge has been organized as algorithms, their combinations and system reactions. The novelty of this approach is in the integration of two knowledge sources that are traditionally used in a separated way. Then, each domain and system are defined by introducing the specific knowledge in this representation. Our proposal has addressed the basics for high-level knowledge representation and defined a model that suits the needs of many applications. This model has been constructed on a neutral framework that is not restricted to any specific analysis tool for its extraction. The proposed modeling is useful for, among others, both developing query-based systems and efficient video analysis applications. Moreover, it can be easily extended to incorporate new knowledge sources.

Secondly, the proposed representation is used to develop video analysis systems showing its effectiveness in the composition of analysis workflows and the guidance of the video event recognition. In chapter 3, we have presented a generic, scalable and distributed framework for video analysis intended for research, prototyping and services deployment purposes. Compared to current related literature, we have proposed a novel modular approach to solve the video

analysis problems of a specific domain by selecting and ordering the optimum tools to compose the analysis workflow. It reduces the complexity for developing video analysis systems that usually require high-specialized experts and, therefore, it increases the re-usability of existing algorithms. Besides, it allows to isolate the design phases in the domain-knowledge-related and algorithmic-related parts. Hence, this design is simplified as domain experts and algorithm designers can focus their efforts in the development of more accurate models or algorithms. Later on, chapter 4 has introduced a knowledge-based approach for reliable video event recognition. Here, the semantic representation is used to define a layered recognition structure based on the event complexity. Each event is tackled with different analysis strategies according to its characteristics. Moreover, the uncertainty of the low-level analysis is considered within the layers of the structure. The novelty of this proposal regards in that it takes advantage of the accuracy of probabilistic approaches as well as the descriptive capabilities of semantic-based approaches. An additional contribution is to provide a formal connection between expert knowledge and recognition models. where the events defined under the proposed representation model can be translated to appropriate recognition structures. The experiments showed that it successfully recognized five human-object interactions. However, an accuracy decrease is observed in high complex situations (e.g., crowded scenarios) due to the limitations of the blob-based analysis.

Thirdly, this thesis has investigated the relatively unexploited field of feedback-based video analysis that allows to control the performance. In chapter 5, we have proposed a feedback processing scheme based on variable level of detail (LoD) for the analysis performed and complexity estimation (CE) for input and output data. This scheme changes its LoD to achieve a desired performance level. Moreover, existing approaches that can be applied to the components of this scheme have been identified. Compared to the traditional feed-forward approach, it allows to deal with unexpected conditions and to study the relations between the stages of complex systems. Afterwards, chapter 6 has studied the application of feedback to the stages of a common video event recognition system. A system manager has been included for coordinating the implemented feedback strategies. Experiments showed that the feedback-based system improved the initial event detection results increasing the precision (reducing the false event detection) whereas slightly decreasing the recall (missing few events). On the other hand, a high computational cost reduction is achieved due to the use of the appropriate LoDs according to the data complexity. These findings may be very useful in several real situations such as the inclusion of additional recognition capabilities maintaining the same computational cost and the variation of the analysis effort on the processing units of a multi-camera setting where the data has high complexity (e.g., cameras with more activity).

Finally, this thesis has concentrated on the main difficulty observed in the use of the feedback scheme: the output quality estimation of a processing stage. We have focused on the evaluation without ground-truth as manual annotations are expensive to produce and their use is not

feasible for on-line performance analysis. Besides its use in the feedback loop, the evaluation without ground-truth presents several advantages such as algorithm ranking over large datasets and suitability for automatic control of online segmentation (self-tuning). In particular, we have studied the most common stages in video analysis: foreground segmentation and object tracking (chapter 7). Both studies propose taxonomies for organizing current literature and compare the most representative approaches on public available datasets. Then, we gave a recommendation on which approach performs better under specific characteristics of the video data. The obtained results showed that the selected approaches have an intermediate accuracy as compared to ground-truth ones. In particular, the evaluation of foreground segmentation heavily depends on the sequence properties (e.g., background motion and texture, foreground velocity and size). On the other hand, the evaluation of object tracking should be faced by combining different approaches. However, this combination still obtained intermediate accuracy. In conclusion, further studies are needed for reliable evaluation of output quality without ground-truth data for foreground segmentation and object tracking. In addition, we have proposed a novel approach for the estimation of track quality without ground-truth (chapter 8). It is based on the uncertainty of the tracking filter and the reverse tracking approach for quality estimation. Unlike the current literature, it identifies the correct tracker operation for estimating its quality instead of a simple computation of statistics derived from tracking data. It establishes whether the tracker is locked on to the right target, locked on to a distractor (e.g., background) or scanning the image looking for the target. Then, track quality estimation is computed for the successful tracker operation by means of another tracker in reverse direction that avoids the exponential computational cost of the reverse-tracking approach by controlling the length of the reverse analysis. It allows to estimate the track quality in long video sequences where the tracker might fail several times. This approach is demonstrated in the particle filter framework and is valid for any multi-hypothesis tracker that uses some form of uncertainty related to the spread of the hypothesis.

9.2 Future Work

Based on the results and discussions of this thesis, we plan the following future research lines:

- Extension of the semantic representation. In chapter 2, the presented approach only considers high level information. We propose to introduce low and mid-level aspects. For example, we can incorporate the pixel ontology defined by [Garcia and Bescos, 2008] to model the pixel status or a mid-level description to separate specific domain knowledge from the low and mid level analysis (like in [Duan et al., 2003] for sports video analysis). These low and mid-level descriptions allow to define better the link between the raw data and the high-level concepts for developing more efficient and portable algorithms.

- Intra-blob analysis for event recognition. As mentioned in chapters 4 and 6, the recognition of events is not fully solved for complex scenarios such as crowded places. Multiple occlusions and groups of people reduce the reliability of the current recognition methods as they assume complete observable persons on a one-to-one blob correspondence basis. Further improvements can be achieved by using new sets of features to segment blobs containing multiple persons (such as the modeling of motion patterns [Simon et al., 2008]).

- Automatic parameter learning for the feedback scheme. We propose to explore efficient methods for learning the optimum thresholds to change the LoD of the feedback scheme of chapter 5. For each processing stage, this research will run all the available LoDs, compute the output quality with annotated data, study their correlation with the output quality estimation not-based on ground-truth and select the right LoD with the highest correlation value. Then, the optimum LoD for each time instant should be used to learn the thresholds for LoD increase and decrease.

- Feedback-based foreground segmentation and tracking. This thesis only considered feedback use from a system viewpoint without implying an increase in the performance of each processing stage (as the aim is the overall system performance). Hence, we propose to apply the feedback scheme of chapter 5 to the foreground segmentation and tracking stages for increasing their robustness. Starting from the initial proposals of chapter 6, their combination with the output quality estimators studied in chapter 7 should be investigated. Although similar approaches already exist in the current literature (such as the tracker switching based on reverse evaluation [Pan et al., 2009b]), they share the limitations of the employed quality estimators for their application in real-world settings.

- Quality estimation without ground-truth for foreground segmentation. We propose to continue the study initiated in chapter 7 for evaluating foreground segmentation without ground-truth data. In particular, three lines of research are suggested for improving the selected contrast-based approach. Firstly, the investigation on adaptive parameter estimation to avoid the limitation on fixed values. Secondly, an improvement can be achieved by revisiting the combination method of the pixel-level difference values. Finally, the use of measures based on entropy [Zhang et al., 2008] should be considered.

- Improve the quality estimator for object tracking. We propose to investigate on adaptive thresholding techniques and the fusion of multiple trackers based on track quality. For real-time deployment, we suggest to research on techniques not-based on reverse tracking as its computational cost has an exponential increase. However, the structure of the presented approach in chapter 8 should be preserved as the identification of the successful tracker operation is considered a critical factor for estimating track quality and it runs in real-time.

Part V

Appendixes

Appendix A

Publications

The following publications have been produced in association with this thesis (listed by chapters):

- Representation of video event semantics (chapter 2)
  - J. C. SanMiguel, J. M. Martínez, and A. García. An ontology for event detection and its application in surveillance video. In Proc. of the IEEE Int. Conf. on Advanced Video and Signal based Surveillance, pages 220-225, Genoa (Italy), 2-4 September 2009 (http://dx.doi.org/10.1109/AVSS.2009.28).
- A self-configurable framework for video analysis (chapter 3)
  - J. C. SanMiguel, J. Bescós, J. M. Martínez, and A. Garcia. Diva: A distributed video analysis framework applied to video-surveillance systems. *In Proc. of IEEE Int. Workshop on Image Analysis for Multimedia Interactive Services*, pages 207-210, 7-9 May 2008 (http://dx.doi.org/10.1109/WIAMIS.2008.29).
  - J. C. SanMiguel and J. M. Martínez. Dynamic video surveillance systems guided by domain ontologies. *In Proc. of the Int. Conf. on Imaging for Crime Detection and Prevention*, pages 1-6, London (UK), 3 December 2009 (http://dx.doi.org/10.1049/ic.2009.0261).
  - J. C. SanMiguel and J. Martinez. A semantic-guided and self-configurable video analysis framework. *Submitted to Machine Vision and Applications*, 2011 (in second review).
- Recognition of single-view human-related video events (chapter 4)
  - J. C. SanMiguel, M. Escudero-Viñolo, J. M. Martínez, and J. Bescós. Real-time single-view video event recognition in controlled environments. In Proc. of the Int.

Workshop on Content-Based Multimedia Indexing, Madrid (Spain), pages 91-96, 14-16 June 2011 (http://dx.doi.org/10.1109/CBMI.2011.5972527).

- J. C. SanMiguel and J. Martinez. A semantic-based probabilistic approach for real-time video event recognition. *Submitted to* Computer *Vision and Image Understanding*, 2011 (in second review).

- Feedback-based analysis model and its application to event detection (chapters 5 and 6)
  - J. C. SanMiguel and J. M. Martínez. Shadow detection in video surveillance by maximizing agreement between independent detectors. *In Proc. of the IEEE Int. Conf. on Image Processing*, pages 1141-1444, Cairo (Egypt), 7-10 November 2009. (http://dx.doi.org/10.1109/ICIP.2009.5413532).
  - J. C. SanMiguel and J. M. Martínez. Use of feedback strategies in the detection of events for video surveillance. *To appear in IET Computer Vision,* 2011.

- On quality estimation without ground-truth for foreground segmentation and tracking (chapter 7)
  - J. C. SanMiguel and J. M. Martínez. On the evaluation of background subtraction algorithms without ground-truth. *In Proc. of the IEEE Int. Conf. on Advanced Video and Signal based Surveillance*, pages 180-187, Boston (USA), 1-3 September 2010 (http://dx.doi.org/10.1109/AVSS.2010.6).
  - J. C. SanMiguel, A. Cavallaro, and J. M. Martínez. Evaluation of on-line quality estimators for object tracking detectors. *In Proc. of the IEEE Int. Conf. on Image Processing*, pages 825-828, Hong Kong (China), 7-10 November 2010 (http://dx.doi.org/10.1109/ICIP.2010.5653449).

- On-line video tracker evaluation using adaptive reverse tracking (chapter 8)
  - J. C. SanMiguel, A. Cavallaro, and J. M. Martínez. Adaptive on-line performance evaluation of video trackers. *Submitted to IEEE Trans. on Image Processing*, 2010.

Electronic versions are available at the following URL:
http://www-vpu.eps.uam.es/webvpu/gti/user/12/

Appendix B

Logros, conclusiones y trabajo futuro

B.1 Resumen de logros y principales conclusiones

Esta tesis ha estudiado la aplicación del paradigma de Visión Cognitiva por Computador al análisis de vídeo. El objetivo es incrementar la robustez de un sistema de detección de eventos para optimizar su uso y tratar datos inesperados. En particular, se han explorado cuatro áreas relacionadas con la descripción de conocimiento (capítulo 2), análisis estructurado (capítulos 3 y 4), uso de realimentación (capítulos 5 y 6) y auto-evaluación (capítulos 7 y 8).

En primer lugar, nos hemos centrado en proporcionar una representación estructurada del conocimiento relacionado con el dominio de aplicación y el sistema de análisis (capítulo 2). Conocimiento del dominio ha sido descrito en términos de objetos, eventos y contexto mientras que el sistema esta organizado en algoritmos, sus combinaciones y posibles reacciones. La novedad de esta aproximación esta en la integración de dos tipos de conocimiento que tradicionalmente se utilizan de manera separada. Después, cada dominio y sistema son definidos mediante la introducción del conocimiento especifico. Nuestra aproximación proporciona los fundamentos para representar conocimiento de alto nivel que encaja en las necesidades de muchas aplicaciones. El modelo ha sido construido de manera independiente a las herramientas de análisis a utilizar. La propuesta es útil para definir sistemas eficientes de análisis e indexado de vídeo. Adicionalmente, puede ser fácilmente extendida para incorporar nuevas fuentes de conocimiento.

En segundo lugar, dicha representación es utilizada para desarrollar sistemas de análisis de vídeo demostrando su efectividad para componer flujos de trabajo y guiar el reconocimiento de eventos. En el capítulo 3, se ha presentado una arquitectura genérica, escalable y distribuida que permite la investigación, prototipado y desarrollo de sistemas. Comparado con la literatura actual, se ha propuesto una aproximación novedosa para resolver el problema de análisis en un dominio especifico mediante la selección y ordenado de las herramientas óptimas para componer el flujo de trabajo. Se ha reducido la complejidad del desarrollo de sistemas de análisis que normalmente requieren expertos cualificados y se ha incrementado la reusabilidad de las

herramientas existentes. Además, la propuesta permite aislar las etapas de diseño relacionadas con el conocimiento y algoritmia. Así pues, el diseño se simplifica y permite que los expertos puedan concentrar sus esfuerzos en una u otra tarea. Después, el capítulo 4 ha introducido un reconocimiento de eventos robusto basado en conocimiento. La representación semántica se utiliza para definir una estructura de reconocimiento por capas basada en la complejidad de los eventos. Por lo tanto, cada evento es detectado con diferentes estrategias de análisis. Además, la incertidumbre del análisis de bajo nivel es considerada en las capas de la estructura. La novedad de esta aproximación reside en la combinación de la precisión de las aproximaciones probabilistas y la capacidad descriptiva de las aproximaciones semánticas. Una contribución adicional es la de formalizar la conexión entre conocimiento y sus métodos de reconocimiento. Así pues, los eventos definidos con el modelo propuesto pueden reconocidos mediante los métodos más apropiados. Los experimentos mostraron que el sistema reconoció satisfactoriamente cinco interacciones persona-objeto. No obstante, un descenso de la precisión se observó en situaciones muy complejas (e.g., escenario poblados) debido a las limitaciones del análisis a nivel de *blob*.

En tercer lugar, esta tesis ha investigado el campo relativamente poco explorado del análisis de vídeo realimentado que permite controlar el rendimiento. En el capítulo 5, se ha propuesto una esquema de procesado realimentado basado en un niveles de detalle para el análisis (LoD) y la estimación de la complejidad para los datos de entrada y de salida (CE). Este esquema cambia su LoD para alcanzar el rendimiento deseado. Además, aproximaciones existentes que pueden ser aplicadas a este esquema han sido identificadas. Comparado con la aproximación prealimentada, nuestra propuesta permite tratar con datos inesperados y estudiar las relaciones entre las etapas de un sistema complejo. Después, el capítulo 6 ha estudiado su aplicación a las etapas de un sistema tradicional de reconocimiento de eventos. Un controlador del sistema se ha introducido para coordinar las estrategias realimentadas e implementadas. Los experimentos muestran que el sistema realimentado mejoró la detección inicial incrementando la precisión (reduciendo la tasa de falsas alarmas) y reduciendo ligeramente el *recall* (perdiendo detecciones correctas). Por otro lado, una reducción drástica del coste computacional fue alcanzada debido al uso de los LoDs apropiados a la complejidad de la escena. Estos resultados pueden ser de gran utilidad en dos situaciones: para incluir métodos de análisis mientras se mantiene el mismo coste computacional y para variar el esfuerzo de análisis de las unidades de procesado de un sistema distribuido (e.g., centrar el esfuerzo en cámaras con alta actividad).

Finalmente, esta tesis se ha concentrado en la principal dificultad observada en el uso de realimentación: la estimación de la calidad de los datos generados por una etapa de análisis. Se ha centrado la atención en la evaluación sin datos anotados pues las anotaciones manuales son difíciles de generar y su uso no es posible para evaluar el rendimiento en sistemas *on-line*. Además de su uso en el bucle realimentado, la evaluación sin datos anotados presenta varias ventajas como: no requiere anotaciones, ranking de algoritmos sobre grandes bases de datos y

auto-ajuste de parámetros. En particular, se han estudiado dos etapas comunes en el análisis de vídeo: segmentación de primer plano y seguimiento de objetos (capítulo 7). La novedad de ambos estudios radica en la creación de taxonomías para la literatura existente y la comparativa de las aproximaciones más relevantes sobre conjuntos de prueba públicos. Después, se proporciona una recomendación de que aproximación funciona mejor en base a las características de la secuencia de vídeo. Los resultados obtenidos exhibieron que las aproximaciones seleccionadas presentan una precisión intermedia comparadas con la evaluación basada en anotaciones. En particular, la evaluación de segmentación de primer plano presenta una gran dependencia con las características de la secuencia de vídeo (e.g., movimiento y textura del fondo, velocidad y tamaño del primer plano). La evaluación del seguimiento de objetos debe ser encarada mediante la combinación de aproximaciones. No obstante, esta combinación sigue obteniendo resultados intermedios. En conclusión, se necesitan más estudios para proporcionar estimaciones fiables de la calidad de salida sin utilizar datos anotados. Adicionalmente, se ha propuesto una aproximación novedosa para evaluar el seguimiento de objetos sin anotaciones (capítulo8). Se basa en el uso de la incertidumbre del filtro y la aproximación de análisis inverso para estimar la calidad. De manera opuesta a la literatura actual, se identifican los instantes temporales en los que el algoritmo esta operando correctamente en lugar de calcular estadísticos derivados de los datos de seguimiento. Permite establecer si el algoritmo esta centrado en el objecto correcto, en un objeto erróneo o esta buscando en la imagen el objeto. Después, la estimación de calidad se realiza para los instantes de operación correcta del algoritmo mediante el uso de otro algoritmo aplicado en la dirección inversa que evita la dependencia exponencial del coste computacional (mediante la adaptación de la longitud del análisis inverso). Por lo tanto, permite su uso en secuencias largas donde se espera que el algoritmo falle varias veces. Esta aproximación se ha demostrado en el marco del filtro de partículas y es válida para cualquier algoritmo de seguimiento multi-hipótesis que utilice alguna forma de incertidumbre relacionada con la separación de dichas hipótesis.

B.2 Trabajo futuro

Basándose en los resultados de esta tesis, se proponen las siguientes extensiones:

- Extensión de la representación semántica. En el capítulo 2, el modelo solamente considera información de alto nivel. Se propone introducir aspectos de bajo y medio nivel. Por ejemplo, se puede incorporar la ontología de píxel propuesta por [Garcia and Bescos, 2008] para modelar el estado del píxel o la descripción de nivel medio para separar conocimiento del dominio de las etapas de análisis (como en [Duan et al., 2003] para el análisis de vídeos de deportes). Estas descripciones permiten definir mejor el enlace entre los datos y los conceptos de alto nivel para el desarrollo de algoritmos más eficientes y portables.

- Análisis *intra-blob* para reconocimiento de eventos. Como se menciona en los capítulos 4 y 6,

dicho reconocimiento no esta solucionado para escenarios complejos. Oclusiones múltiples y grupos de personas reducen la fiabilidad de los métodos actuales de reconocimiento debido a que asumen personas completamente observables que se corresponden univocamente con un *blob*. Mejoras se pueden alcanzar mediante el uso de nuevas características que permitan segmentar *blobs* conteniendo múltiples personas (como el uso de patrones de movimiento [Simon et al., 2008]).

- Aprendizaje automático de los parámetros del esquema realimentado. Se propone explorar métodos eficientes de aprendizaje de los umbrales óptimos para el cambio de nivel de detalle de análisis (LoD) del esquema realimentado del capítulo 5. Para cada etapa, este estudio deberá utilizar todos los LoDs disponibles, calcular la calidad de su resultado con datos anotados, analizar su correlación con medidas sin datos anotados y seleccionar aquel LoD con el valor máximo de correlación. Después, estos LoD óptimos deberían ser utilizados para aprender los umbrales óptimos para subir o bajar el LoD.

- Segmentación de primer plano y seguimiento basado en realimentación. Esta tesis solamente ha considerado realimentación desde un punto de vista de sistema sin implicar una mejora de cada etapa. Por lo tanto, se propone aplicar el esquema del capítulo 5 a las etapas de segmentación de primer plano y seguimiento de objetos para incrementar su robustez. Comenzando con las propuestas iniciales del capítulo 6, su combinación con medidas de calidad de salida (capítulo 7) debe ser estudiada. Aunque aproximaciones similares existen en el estado del arte (como la selección del algoritmo de seguimiento en base al análisis en dirección contraria [Pan et al., 2009b]), éstas comparten las limitaciones de los estimadores de calidad empleados para su uso en escenarios reales.

- Estimación de calidad sin datos anotados para segmentación de primer plano. Se propone continuar el estudio iniciado en el capítulo 7 para evaluar segmentación sin datos anotados. En particular, se sugieren tres líneas de trabajo para mejorar la aproximación seleccionada. Primero, la investigación de métodos para estimar los parámetros de manera adaptativa para evitar la limitación del uso de parámetros fijos. Segundo, mejorar el proceso de combinación de las diferencias a nivel de píxel. Finalmente, el uso de medidas basadas en la entropía [Zhang et al., 2008] debe ser considerado.

- Mejorar el estimador de calidad para seguimiento de objetos. Se propone investigar en técnicas adaptativas de umbralizado y la fusión de múltiple algoritmos basada en calidad. Para desarrollo en tiempo real, se sugiere investigar en técnicas no basadas en el análisis en dirección contraria debido a que presentan un coste computacional con dependencia exponencial. No obstante, la estructura de la propuesta del capítulo 8 debe ser preservada debido a que la identificación del correcto funcionamiento del algoritmo se considera un factor crítico para estimar calidad de seguimiento y funciona en tiempo real.

Glossary

BLOB *Binary Large OBject*

BN *Bayesian Networks*

CCA *Connected Component Analysis*

CCV *Cognitive Computer Vision*

CFG *Context-Free Grammar*

CORBA *Common Object Request Broker Architecture*

CPU *Central Processing Unit*

DAG *Directed Acyclic Graph*

DBN *Dynamic Bayesian Networks*

DL *Description Logic*

FSM *Finite State Machine*

GMM *Gaussian Mixture Model*

GT *Ground-Truth*

HMM *Hidden Markov Model*

HSV *Hue Saturation Value color model*

JPEG *Joint Picture Experts Group*

KDE *Kernel Density Estimator*

MPEG *Moving Picture Experts Group*

MoSIFT *Motion Scale-Invariant Feature Transform*

NGT *Not based on Ground-Truth*

NN *Neural Network*

OWL *Ontology Web Language*

PN *Petri Net*

POI *Point Of Interest*

ROI *Region Of Interest*

RGB *Red Green Blue color model*

RTSP *Real-Time Streamming Protocol*

SIFT *Scale-Invariant Feature Transform*

SM *Semantic Model*

SOAP *Simple Object Access Protocol*

SURF *Speeded-Up Robust Features*

SVM *Support Vector Machine*

SyM *Syntactic Modeling*

SWRL *Semantic Web Rule Language*

URL *Universal Resource Locator*

VERL *Video Event Representation Language*

XML *eXtensible Markup Language*

Bibliography

C. Achard, X. Qu, A. Mokhber, and M. Milgram. A novel approach for recognition of human actions with semi-global features. *Machine Vision and Applications*, 19(1):27–34, Jan. 2008. [Cited on pages 56 and 58.]

J. K. Aggarwal and M. S. Ryoo. Human activity analysis : A review. *ACM Computing Surveys*, 43(3), Apr. 2011. [Cited on pages 13, 54, and 55.]

U. Akdemir, P. Turaga, and R. Chellappa. An ontology based approach for activity recognition from video. In *Proc. of the ACM Int. Conf. on Multimedia*, pages 709–712, Vancouver (Canada), 26-31 Oct. 2008. [Cited on pages 14 and 57.]

M. Albanese, R. Chellappa, V. Moscato, A. Picariello, V. S. Subrahmanian, P. Turaga, and O. Udrea. A constrained probabilistic petri net framework for human activity detection in video. *IEEE Trans. on Multimedia*, 10(8):1429–443, December 2008. [Cited on pages 57, 58, and 63.]

S. Ali and M. Shah. Human action recognition in videos using kinematic features and multiple instance learning. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 32(2):288–303, Feb. 2010. [Cited on page 54.]

J. Allen. Actions and events in interval temporal logic. *Journal of Logic and computation*, 4(5):531–579, July 1994. [Cited on pages 13, 14, and 21.]

K. R. Apt. *Principles of Constraint Programming*. Cambridge University Press, 2003. [Cited on pages 32 and 45.]

A. Artikis, M. Sergot, and G. Paliouras. A logic-programming approach to activity recognition. In *Proc. of ACM Int. Workshop on Events in Multimedia*, pages 3–8, Firenze (Italy), 25-29 Oct. 2010. [Cited on page 13.]

A. Avanzi, F. Bremond, C. Tornieri, and M. Thonnat. Design and assessment of an intelligent activity monitoring platform. *EURASIP Journal on Applied Signal Processing*, pages 2359–2374, 2005. [Cited on pages 33 and 35.]

D. Ayers and M. Shah. Monitoring human behavior from video taken in an office environment. *Image and Vision Computing*, 19(12):833–846, December 2001. [Cited on pages 54, 56, and 57.]

V. Badrinarayanan, P. Perez, F. Le Clerc, and L. Oisel. On uncertainties, random features and object tracking. In *Proc. of IEEE Int. Conf. on Image Processing*, pages 61–64, San Antonio (TX, USA), 16-19 Sept. 2007a. [Cited on page 124.]

V. Badrinarayanan, P. Perez, F. Le Clerc, and L. Oisel. Probabilistic color and adaptive multi-feature tracking with dynamically switched priority between cues. In *Proc. of IEEE Int. Conf. on Computer Vision*, pages 1–8, Rio de Janeiro (Brasil), 14-21 Oct. 2007b. [Cited on pages 124, 129, 142, 143, 144, and 145.]

A. Bagdanov, A. Del-Bimbo, F. Dini, and W. Nunziati. Adaptive uncertainty estimation for particle filter-based trackers. In *Proc. of Int. Conf. on Image Analysis and Processing*, pages 331–336, Modena (Italy), 10-14 Sept. 2007. [Cited on page 124.]

L. Bai, S. Lao, G. Jones, and A. Smeaton. Video semantic content analysis based on ontology. In *Proc. of Int. Machine Vision and Image Processing Conf.*, pages 117–124, Maynooth (Ireland), 5-7 Sept. 2007. [Cited on pages 15, 16, and 34.]

L. Ballan, M. Bertini, A. Del Bimbo, and G. Serra. Semantic annotation of soccer videos by visual instance clustering and spatial/temporal reasoning in ontologies. *Multimedia Tools and Applications*, 48(2):313–337, Feb. 2010. [Cited on page 13.]

L. Ballan, M. Bertini, A. Del Bimbo, L. Seidenari, and G. Serra. Event detection and recognition for semantic annotation of video. *Multimedia Tools and Applications*, 51(1):279–302, Jan. 2011. [Cited on pages 13 and 54.]

A. Bayona, J. C. SanMiguel, and J. M. Martinez. Comparative evaluation of stationary foreground object detection algorithms based on background subtraction techniques. In *Proc. of the IEEE Int. Conf. on Advanced Video and Signal based Surveillance*, pages 25–30, Genova (Italy), 2-4 Sept. 2009. [Cited on page 48.]

A. Bayona, J. C. SanMiguel, and J. M. Martinez. Stationary foreground detection using background subtraction and temporal difference in video surveillance. In *Proc. of IEEE Int. Conf. on Image Processing*, pages 4657–4660, 26-29 Sept. 2010. [Cited on pages 48 and 69.]

Y. Benezeth, P. Jodoin, B. Emile, H. Laurent, and C. Rosenberger. Comparative study of background subtraction algorithms. *Journal of Electronic Imaging*, 19(3):11, July 2010. [Cited on page 119.]

A. Binotto, E. de Freitas, C. Pereira, A. Stork, and T. Larsson. Real-time task reconfiguration support applied to an uav-based surveillance system. In *Proc. of the Int. Multiconf. on Computer Science and Information Technology*, pages 581–588, Wisia, 20-22 Oct. 2008. [Cited on page 34.]

J. Bins, T. List, R. B. Fisher, and D. Tweed. An intelligent and task-independent controller for video sequence analysis. In *Proc. of the Int. Workshop on Computer Architecture for Machine Perception*, pages 172–177, Palermo (Italy), 4-6 July 2005. [Cited on pages 34 and 90.]

J. Black, T. Ellis, and P. Rosin. A novel method for video tracking performance evaluation. In *Proc. of IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance*, pages 125–132, Nice (France), Oct. 2003. [Cited on pages 123, 124, 147, and 148.]

A. Bobick and J. Davis. The recognition of human movement using temporal templates. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(3):257–267, Mar. 2001. [Cited on page 54.]

A. Bobick and A. D. Wilson. A state-based approach to the representation and recognition of gesture. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(12):1325–1337, December 1997. [Cited on pages 13, 57, and 58.]

R. Brachman. Systems that know what they're doing. *IEEE Intelligent Systems*, 17(6):67–71, Nov./December 2002. [Cited on page 3.]

A. Caporossi, D. Hall, P. Reignier, and J. L. Crowley. Robust visual tracking from dynamic control of processing. In *Proc. of the IEEE Workshop on Performance Evaluation for tracking and Surveillance*, Prague (Czech Republic), 10 May 2004. [Cited on pages 48 and 100.]

C. Carincotte, X. Desurmont, B. Ravera, F. Bremond, J. Orwell, S. Velastin, J. Odobez, B. Corbucci, J. Palo, and J. Cernocky. Toward generic intelligent knowledge extraction from video and audio: the eu-funded caretaker project. In *Proc. of the Int. Conf. on Imaging for Crime Detection and Prevention*, pages 1–6, London (UK), 13-14 June 2006. [Cited on pages 33 and 35.]

E. Carmona, M. Rincon, M. Bachiller, J. Martinez-Cantos, R. Martinez-Tomas, and J. Mira. On the effect of feedback in multi level representation spaces for visual surveillance tasks. *Neurocomputing*, 72 (4-6):916–927, Jan. 2009. [Cited on pages 92 and 99.]

C. Castel, L. Chaudron, and C. Tessier. What is going on? a high level interpretation of sequences of images. In *Proc. of IEEE Eur. Conf. on Computer Vision, Workshop on Conceptual Descriptions from Images*, Cambridge(UK), Apr. 1996. [Cited on page 56.]

A. Cavallaro and T. Ebrahimi. Interaction between high-level and low-level image analysis for semantic video object extraction. *EURASIP Journal on Applied Signal Processing*, 2004(6):786–797, Jan. 2004. [Cited on page 91.]

A. Cavallaro, O. Steiger, and T. T. Ebrahimi. Semantic video analysis for adaptive content delivery and automatic description. *IEEE Trans. on Circuits and Systems for Video Technology*, 15(10):1200–1209, Oct. 2005. [Cited on pages 48, 68, 73, 93, 100, 102, 133, 138, 139, 140, and 141.]

S. Chabrier, B. Emile, C. Rosenberger, and H. Laurent. Unsupervised performance evaluation of image segmentation. *EURASIP Journal on Applied Signal Processing*, 1(1):1–12, Jan. 2006. [Cited on page 121.]

D. Chau, F. Bremond, and M. Thonnat. Online evaluation of tracking algorithm performance. In *Proc. of Int. Conf. on Imaging for Crime Prevention and Detection*. London (UK), December 2009. [Cited on pages 123, 124, 147, and 148.]

D. Chen, J. Yang, and H. Wactlar. Towards automatic analysis of social interaction patterns in a nursing home environment from video. In *Proc. of the ACM SIGMM Int. Workshop on Multimedia Information Retrieval*, pages 283–290, New York (USA), 10-16 Oct. 2004. [Cited on page 13.]

M. Chen and A. Hauptmann. Mosift: Recognizing human actions in surveillance videos. Technical Report CMU-CS-09-161, Carnegie Mellon University, 2009. [Cited on page 55.]

P. Chen and P. Popovich. *Correlation: Parametric and non-parametric measures.* Thousand Oaks, 2002. [Cited on pages 105, 131, and 160.]

S. Cheung and C. Kamath. Robust background subtraction with foreground validation for urban traffic video. *EURASIP Journal on Applied Signal Processing*, 1(1):2330–2340, Jan. 2005. [Cited on page 122.]

H. I. Christensen and H.-H. Nagel. *Cognitive Vision Systems: Sampling the Spectrum of Approaches.* Springer-Verlag, 2006. [Cited on page 3.]

R. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(10):1631–1643, Oct. 2005. [Cited on pages 123, 148, and 157.]

P. Correia and F. Pereira. Stand-alone objective segmentation quality evaluation. *EURASIP Journal on Applied Signal Processing*, 4(1):761–774, Jan. 2002. [Cited on page 123.]

P. Correia and F. Pereira. Objective evaluation of video segmentation quality. *IEEE Trans. on Image Processing*, 12(2):186–200, Feb. 2003. [Cited on pages 120 and 121.]

N. Cuntoor, B. Yegnanarayana, and R. Chellappa. Interpretation of state sequences in hmm for activity representation. In *Proc. of IEEE Int. Conf. Acoustics, Speech and Signal Processing*, volume 2, pages 709–712, Philadelphia (USA), 18-23 Mar. 2005. [Cited on page 56.]

F. Dadgostar and A. Sarrafzadeh. An adaptive real-time skin detector based on hue thresholding: A comparison on two motion tracking methods. *Pattern Recognition Letters*, 27(1):1342–1352, Sept. 2006. [Cited on pages 48 and 70.]

S. Dasiopoulou, V. Mezaris, I. Kompatsiaris, V.-K. Papastathis, and M. Strintzis. Knowledge-assisted semantic video object detection. *IEEE Trans. on Circuits and Systems for Video Technology*, 15(10): 1210–1224, Oct. 2005. [Cited on pages 13, 14, 34, 35, and 42.]

H. Detmold, A. Dick, K. Falkner, and R. Morrison. Scalable surveillance software architecture. In *Proc. of IEEE Int. Conf. on Advanced Video and Signal based Surveillance*, pages 1–6, Sidney (Australia), 22-24 Nov. 2006. [Cited on page 33.]

D. Doermann and D. Mihalcik. Tools and techniques for video performances evaluation. In *Proc. of IEEE Int. Conf. on Pattern Recognition*, pages 167–170, Cambrigde (UK), 23-26 Aug. 2000. [Cited on pages 28, 74, and 112.]

A. Doulamis. Dynamic tracking re-adjustment: a method for automatic tracking recovery in complex visual environments. *Multimedia Tools and Applications*, 50(1):49–73, Jan. 2010. [Cited on page 123.]

L.-Y. Duan, M. Xu, T.-S. Chua, Q. Tian, and C.-S. Xu. A mid-level representation framework for semantic sports video analysis. In *Proc. of the ACM Int. Conf. on Multimedia*, pages 33–44, 2003. [Cited on pages 173 and 181.]

A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. In *Proc. of IEEE Eur. Conf. on Computer Vision*, pages 751–767, Dublin (Ireland), 26 June - 1 Jule 2000. [Cited on pages 94, 133, 138, 139, 140, and 141.]

S. Y. Elhabian, K. M. El-Sayed, and S. H. Ahmed. Moving object detection in spatial domain using background removal techniques - state-of-art. *Recent Patents on Computer Science*, 1(1):32–54, Jan. 2008. [Cited on page 78.]

C. Erdem, E. Sankur, and A. Tekalp. Performance measures for video object segmentation and tracking. *IEEE Trans. on Image Processing*, 13(7):937–951, July 2004a. [Cited on pages 48, 97, 121, 123, 125, 126, and 148.]

C. Erdem, A. Tekalp, and B. Sankur. Video object tracking with feedback of performance measures,. *IEEE Trans. on Circuits and Systems for Video Technology*, 13(4):310–324, Apr. 2004b. [Cited on pages 91 and 123.]

M. Farrell, B. Walthall, A. Vaswani, F. Thompson, and M. DeVore. Rapidly deployable distributed video surveillance system for resource constrained applications. In *Proc. of the IEEE Conf. on System and Information Engineering Design*, pages 1–4, Virginia (USA), 27 Apr. 2007. [Cited on page 33.]

C. Fernandez. *Understanding Image Sequences: the Role of Ontologies in Cognitive Vision*. PhD thesis, Universitat Autonoma de Barcelona, Spain, July 2010. [Cited on pages 4 and 12.]

C. Fernandez, P. Baiget, X. Roca, and J. Gonzalez. Interpretation of complex situations in a semantic-based surveillance framework. *Signal Processing: Image Communication*, 23(7):554–569, Aug. 2008. [Cited on pages 14 and 16.]

V. Fernandez-Carbajales, J. M. Martinez, and M. Garcia. Robust people detection by fusion of evidence from multiple methods. In *Proc. of IEEE Int. Workshop on Image Analysis for Multimedia Interactive Services*, pages 55–58, 7-9 May 2008. [Cited on pages 48, 100, and 107.]

A. Francois, R. Nevatia, J. Hobbs, and R. Bolles. VERL: an ontology framework for representing and annotating video events. *IEEE Multimedia*, 12(4):76–86, Oct./December 2005. [Cited on pages 4, 15, 16, 19, and 22.]

G. Franklin, D. Powell, and A. Emami-Naein. *Feedback Control of Dynamic Systems*. Prentice Hall, 2009. [Cited on pages 87, 88, and 94.]

F. Fusier, V. Valentin, F. Bremond, M. Thonnat, M. Borg, D. Thirde, and J. Ferryman. Video understanding for complex activity recognition. *Machine Vision and Applications*, 18(3):167–188, May 2007. [Cited on pages 14, 16, 57, and 58.]

X. Gao, T. B. amd F. Coetzee, and V. Ramesh. Error analysis of background adaptation. In *Proc. of the IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 503–510, Hilton Head, SC (USA), 13-15 June 2000. [Cited on page 120.]

A. Garcia and J. Bescos. Video object segmentation based on feedback schemes guided by a low-level scene ontology. In *Proc. of Advanced Concepts for Intelligent Vision Systems*, pages 322–333, 20-24 Oct. 2008. [Cited on pages 91, 122, 173, and 181.]

A. Garcia-Martin and J. M. Martinez. Robust real time moving people detection in surveillance scenarios. In *Proc. of IEEE Int. Conf. on Advanced Video and Signal based Surveillance*, pages 241–247, Boston (USA), 1-3 Sept. 2010. [Cited on pages 48 and 69.]

B. Georis, M. Maziere, F. Bremond, and M. Thonnat. A video interpretation platform applied to bank agency monitoring. In *Proc. of Int. Conf. on Intelligent Distributed Surveillance Systems*, pages 46–50, London (UK), Feb. 2004. [Cited on pages 14 and 97.]

N. Ghanem, D. DeMenthon, D. Doermann, and L. Davis. Representation and recognition of events in surveillance video using petri nets. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition Workshop*, pages 104–112, Washington DC (USA), 27 june 2004. [Cited on pages 56, 58, and 63.]

L. Goldmann, T. Adamek, P. Vajda, M. Karaman, R. Morzinger, E. Galmar, T. Sikora, N. O'Connor, T. Ha-Minh, T. Ebrahimi, P. Schallauer, and B. Huet. Towards fully automatic image segmentation evaluation. In *Proc. of Advanced Concepts for Intelligent Vision Systems*, pages 566–577, 20-24 Oct. 2008. [Cited on page 122.]

A. Graser and D. Ristic. Feedback structures as a key requirement for robustness: case studies in image processing. In A. Schuster, editor, *Robust Intelligent Systems*. Springer-Verlag, 2008. [Cited on pages 87, 89, and 90.]

B. Greoris, F. Bremond, and M. Thonnat. Real-time control of video surveillance systems with program supervision techniques. *Machine Vision and Applications*, 18(3-4):189–205, Aug. 2007. [Cited on pages 32, 34, and 35.]

A. Hakeem and M. Shah. Ontology and taxonomy collaborated framework for meeting classification. In *Proc. of the IEEE Int. Conf. on Pattern Recognition*, pages 219–222, Cambridge (UK), 23-26 Aug. 2004a. [Cited on page 13.]

A. Hakeem, Y. Sheikh, and M. Shah. $CASE^E$: A hierarchical event representation for the analysis of videos. In *Proc. of American Association of Artificial Intelligence*, pages 263–268, San Jose (USA), 24-29 July 2004b. [Cited on page 15.]

D. Hall. Automatic parameter regulation of perceptual systems. *Image and Vision Computing*, 24(8): 870–881, Aug. 2006. [Cited on pages 31, 34, 35, 90, 97, and 123.]

Z. Han, Q. Ye, and J. Jiao. Online feature evaluation for object tracking using kalman filter. In *Proc. of the IEEE Int. Conf. on Pattern Recognition*, pages 1–4, Tampa (FL, USA), 8-11 Dec. 2008. [Cited on pages 123 and 133.]

M. Harville. A framework for high-level feedback to adaptive, per pixel, mixture-of-gaussian background models. In *Proc. of the IEEE Eur. Conf. on Computer Vision*, pages 543–560, Copenhagen (Denmark), 27 May-2 June 2002. [Cited on pages 90, 97, and 122.]

T. Henry, E. Janapriya, and L. de Silva. An automatic system for multiple human tracking and actions recognition in office environment. In *Proc. of IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pages 45–48, Hong Kong (China), 6-10 Apr. 2003. [Cited on page 54.]

S. Herrero and J. Bescos. Background subtraction techniques: Systematic evaluation and comparative analysis. In *Proc. of Advanced Concepts for Intelligent Vision Systems*, pages 33–42, 28 Sept.-2 Oct. 2009. [Cited on pages 120 and 133.]

S. Hongeng, R. Nevatia, and F. Bremond. Video-based event recognition: activity representation and probabilistic recognition methods. *Computer Vision and Image Understanding*, 96(2):129–162, Nov. 2004. [Cited on pages 14, 16, 56, and 101.]

L. Hotz, B. Neumann, and K. Terzic. High-level expectations for low-level image processing. In *Proc. of the German conf. on Articial Intelligence*, pages 87–94, Kaiserslautern (Germany), 23-26 Sept. 2008. [Cited on page 91.]

M.-K. Hu. Visual pattern recognition by moment invariants. *IRE Trans. on Information Theory*, 8(2): 179–187, Feb. 1962. [Cited on page 69.]

W. Huang and Q. Wu. Human action recognition based on self organizing map. In *Proc. of IEEE Int. Conf. on Acoustics Speech and Signal Processing*, pages 2130 –2133, Dallas (USA), 14-19 Mar. 2010. [Cited on page 55.]

Y. Ivanov and A. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):852–872, Aug. 2000. [Cited on pages 13, 16, and 57.]

E. Izquierdo, A. Katsaggelos, and M. Strintzis. Special issue on audio and video analysis for multimedia interactive services. *IEEE Trans. on Circuits and Systems for Video Technology*, 14(5):569–571, May 2004. [Cited on page 5.]

E. Jaspers, R. Wijnhoven, R. Albers, J. Nesvadba, J. Lukkien, A. Sinitsyn, X. Desurmont, P. Pietarila, J. Palo, and R. Truyen. Candela - storage, analysis and retrieval of video content in distributed systems. In *Proc. of Int. Workshop on Adaptive Multimedia Retrieval*, pages 112–127, Glasgow (UK), 28-29 July 2005. [Cited on pages 33 and 35.]

Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-backward error: Automatic detection of tracking failures. In *Proc. of IEEE Int. Conf. on Pattern Recognition*, pages 2756–2759, Istambul (Turkey), 23-26 Aug. 2010. [Cited on pages 123, 124, 148, and 156.]

R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, R. Bowers, M. Boonstra, V. Korzhova, and J. Zhang. Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data,metrics, and protocol. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 31(2):319 –336, Feb. 2008. [Cited on pages 122 and 147.]

Y. Ke, R. Sukthankar, and M. Hebert. Volumetric features for video event detection. *Int. Journal of Computer Vision*, 88(1):339–362, July 2010. [Cited on page 54.]

E. Kilambi, A. Ribnick, O. Joshi, and N. Papanikolopoulos. Estimating pedestrian counts in groups. *Computer Vision and Image Understanding*, 110(1):43–59, Apr. 2008. [Cited on pages 48 and 69.]

Y. Kompatsiaris and P. Hobson. *Semantic multimedia and ontologies: theory and applications*. Springer-Verlag, 2008. [Cited on page 11.]

O. Kubassova, M. Boesen, and H. Bliddal. General framework for unsupervised evaluation of quality segmentation results. In *Proc. of the IEEE Int. Conf. on Image Processing*, pages 3036–3039, San Diego (USA), 12-15 Oct. 2008. [Cited on page 121.]

P. Kumar, S. Ranganath, H. Weimin, and K. Sengupta. Framework for real-time behavior interpretation from traffic video. *IEEE Trans. on Intelligent Transportation Systems*, 6(1):43–53, Mar. 2005. [Cited on page 56.]

I. Laptev. On space-time interest points. *Int. Journal of Computer Vision*, 64(2-3):107–123, Sept. 2005. [Cited on page 55.]

G. Lavee, A. Borzin, M. Rudzsky, and E. Rivlin. Building petri nets from video event ontologies. In *Proc. of Int. Symposium on Advances in Visual Computing*, pages 442–451, Lake Tahoe (USA), Nov. 26-28 2007. [Cited on pages 64 and 65.]

G. Lavee, E. Rivlin, and M. Rudzsky. Understanding video events: A survey of methods for automatic interpretation of semantic occurences in video. *IEEE Trans. on Systems, Man and Cybernetics - Part C: Applications and reviews*, 39(5):489–504, Sept. 2009. [Cited on pages 12, 13, 54, 55, and 58.]

G. Lavee, M. Rudzsky, and E. Rivlin. Propagating uncertainty in petri nets for activity recognition. In *Proc. of Int. Symposium on Advances in Visual Computing*, pages 706–715, Las Vegas (USA), Nov. 29-1 2010a. [Cited on page 57.]

G. Lavee, M. Rudzsky, E. Rivlin, and A. Borzin. Video event modeling and recognition in generalized stochastic petri nets. *IEEE Trans. on Circuits and Systems for Video Technology*, 20(1):102–118, Jan. 2010b. [Cited on pages 12, 57, and 63.]

S. Lefevre, L. Mercier, V. Tiberghien, and N. Vincent. Multiresolution color image segmentation applied to background extraction in outdoor images. In *Proc. of IST Eur. Con. on Color in Graphics, Image and Vision*, pages 363–367, Poitiers (France), Apr. 2002. [Cited on pages 96 and 102.]

R. Liu, S. Li, X. Yuan, and R. He. Online determination of track loss using template inverse matching. In *Proc. of the Int. Workshop on Visual Surveillance*, Marseille (France), 17 Oct. 2008. [Cited on pages 123, 128, 129, 142, 143, 144, 145, 158, 160, 161, 162, 164, 166, and 167.]

E. Loutas, I. Pitas, and C. Nikou. Entropy-based metrics for the analysis of partial and total occlusion in video object tracking. *IEE Proc. - Vision, Image, and Signal Processing*, 151(6):487–497, June 2004. [Cited on page 124.]

L. Lu, X. Dai, and G. Hager. A particle filter without dynamics for robust 3d face tracking. In *Proc. of the IEEE Int. Conf. on Computer Vision and Pattern Recognition Workshop*, volume 5, pages 70–73, Washington (DC, USA), 27 June- 2 July 2004. [Cited on page 124.]

Q. Luo, X. Kong, G. Zeng, and J. Fan. Human action detection via boosted local motion histograms. *Machine Vision and Applications*, 21(3):377–389, Apr. 2010. [Cited on pages 55 and 58.]

F. Lv, X. Song, V. Wu, B.and Kumar, and R. Nevatia. Left luggage detection using bayesian inference. In *Proc. of IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance*, pages 83–90, New York (USA), 18 June 2006. [Cited on pages 56 and 62.]

E. Maggio and A. Cavallaro. *Video tracking: theory and practice*. Wiley, 2011. [Cited on pages 119, 122, 147, and 156.]

E. Maggio, F. Smerladi, and A. Cavallaro. Adaptive multifeature tracking in a particle filtering framework,. *IEEE Trans. on Circuits and Systems for Video Technology*, 17(10):1348–1359, Oct. 2007. [Cited on pages 97, 124, 147, 148, 150, 158, 160, 161, 162, 164, 166, and 167.]

N. Maillot, M. Thonnat, and A. Boucher. Towards ontology-based cognitive vision. *Machine Vision and Applications*, 16(1):33–40, Jan. 2004. [Cited on page 32.]

L. Marcerano, F. Oberti, G. Foresti, and C. Regazzoni. Distributed architectures and logical-task decomposition in multimedia surveillance systems. *Proc. of IEEE*, 89(10):1419–1440, 2001. [Cited on pages 31, 33, 34, and 35.]

J. Martinez, R. Koenen, and F. Pereira. Mpeg-7: The generic multimedia content description standard, part 1. *IEEE Multimedia*, 9(2):78–87, Apr./June 2002. [Cited on page 20.]

R. Martinez-Tomas, M. Rincon, M. Bachiller, and J. Mira. On the correspondence between objects and events for the diagnosis of situations in visual surveillance tasks. *Pattern Recognition Letters*, 29(8): 1117–1135, June 2008. [Cited on pages 14, 15, 16, 54, 56, 58, 65, 71, and 101.]

G. Medioni, I. Cochen, F. Bremond, S. Hongeng, and R. Nevatia. Event detection and analysis from video streams. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(8):873–889, Aug. 2001. [Cited on pages 13, 16, 54, 56, and 57.]

A. Mehmet and Z. Choukair. Dynamic, adaptive and reconfigurable systems overview and prospective vision. In *Proc. of IEEE Int. Conf. on Distributed Computing Systems*, pages 84–89, Providence (USA), 18-22 May 2003. [Cited on page 31.]

R. Messing, C. Pal, and H. Kautz. Activity recognition using the velocity histories of tracked keypoints. In *IEEE Proc. of Int. Conf. on Computer Vision*, pages 104–111, Kyoto (Japan), Sept. 29-Oct. 2 2009. [Cited on page 55.]

M. Mirmehdi, P. Palmer, J. Kittler, and H. Davis. Feedback control strategies for object recognition. *IEEE Trans. on Image Processing*, 8(8):1084–1101, Aug. 1999. [Cited on page 90.]

C. Motamed. Motion detection and tracking using belief indicators for an automatic visual-surveillance system. *Image and Vision Computing*, 24(11):1192–1201, Nov. 2006. [Cited on page 123.]

G. Nadarajan. *Semantics and Planning Based Workflow Composition and Execution for Video Processing*. PhD thesis, University of Edinburgh, UK, Aug. 2010. [Cited on pages 34 and 35.]

H.-H. Nagel. Steps toward a cognitive vision system. *AI Magazine*, 25(2):31–50, Summer 2004. [Cited on pages 5, 14, 87, 89, and 99.]

J. Nascimento and J. Marques. Performance evaluation of object detection algorithms for video surveillance. *IEEE Trans. on Multimedia*, 8(4):761–774, Aug. 2006. [Cited on page 121.]

A. Nghiem, F. Bremond, M. Thonnat, and V. Valentin. Etiseo, performance evaluation for video surveillance systems. In *IEEE Conf. on Advanced Video and Signal Based Surveillance*, pages 476–481, London (UK), 5-7 Sept. 2007. [Cited on page 154.]

K. Nickels and S. Hutchinson. Estimating uncertainty in ssd-based feature tracking. *Image and Vision Computing*, 20(1):47–58, Jan. 2002. [Cited on page 124.]

K. Nummiaro, E. Koller-Meier, and E. Van Gool. An adaptive colour-based particle filter. *Image and Vision Computing*, 21(1):99–110, Jan. 2003. [Cited on pages 97, 135, 150, and 158.]

C. O'Conaire, N. O'Connor, E. Cooke, and A. Smeaton. Detection thresholding using mutual information. In *Proc. of Int. Conf. on Computer Vision Theory and Applications*, pages 408–415, Setubal (Portugal), 5-28 Feb. 2006. [Cited on pages 91 and 98.]

C. O'Conaire, N. O'Connor, and A. Smeaton. Detector adaptation by maximising agreement between independent data sources. In *Proc. of the IEEE Int. workshop on Object Tracking and Classification in and Beyond the Visible Spectrum*, pages 1–6, Minneapolis (USA), 22 June 2007a. [Cited on page 122.]

C. O'Conaire, N. O'Connor, and A. Smeaton. Detector adaptation by maximising agreement between independent data sources. In *IEEE Proc. of Int. Workshop on Object Tracking and Classification Beyond the Visible Spectrum*, pages 1–6, Minneapolis (USA), 22 June 2007b. [Cited on pages 91, 96, 98, and 104.]

A. Oerlemans and B. Thomee. Interactive feedback for video tracking using a hybrid maximum likelihood similarity measure. In *Proc. of Human-Computer Interaction*, pages 79–87, Beijing (China), 22-27 July 2007. [Cited on page 90.]

N. Oliver, B. Rosario, and A. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):831–843, Aug. 2000. [Cited on pages 94, 133, 138, 139, 140, and 141.]

P. Pan, F. Porikli, and D. Schonfeld. A new method for tracking performance evaluation based on reflective model and perturbation analysis. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pages 3529–3532, Taipei (Taiwan), 19-24 Apr. 2009a. [Cited on pages 124 and 156.]

P. Pan, F. Porikli, and D. Schonfeld. Recurrent tracking using multifold consistency. In *Proc. of IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance*, Miami (USA), 20-25 June 2009b. [Cited on pages 123, 174, and 182.]

S. Park and J. K. Aggarwal. A hierarchical bayesian network for event recognition of human actions and interactions. *Multimedia Systems*, 10(2):164–179, Aug. 2004. [Cited on pages 56 and 58.]

C. Piciarelli, G. Foresti, and L. Snidaro. Trajectory clustering and its applications for video surveillance. In *Proc. of IEEE Advanced Video-based Signal Surveillance*, pages 40–45, Como (Italy), 15-16 Sept. 2005. [Cited on page 123.]

R. Piroddi and T. Vlachos. A method for single-stimulus quality assessment of segmented video. *EURASIP Journal on Applied Signal Processing*, 1(1):1–22, Jan. 2006. [Cited on page 121.]

E. Polat, M. Yeasin, and R. Sharma. Tracking body parts of multiple people: a new approach. In *Proc. of the IEEE Workshop on Multi-Object Tracking*, pages 35–42, Vancouver (Canada), Aug. 2001. [Cited on page 123.]

R. Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6): 976–990, June 2010. [Cited on page 54.]

R. M. Powers and L. Y. Pao. Power and robustness of a track-loss detector based on kolmogorov-smirnov tests. In *Proc. of American Control Conf.*, pages 3757–3764, Minneapolis (MN, USA), 14-16 June 2006. [Cited on page 124.]

A. Prati, I. Mikic, M. Trivedi, and R. Cucchiara. Detecting moving shadows: Algorithms and evaluation. *IEEE Trans. On Pattern Analysis and Machine Intelligence*, 25(7):918–923, July 2003. [Cited on pages 48, 68, 73, 100, and 104.]

V. Ramesh. Real-time vision at siemens corporate research. In *Proc. of IEEE Int. Conf. on Advanced Video and Signal based Surveillance*, pages 300–305, Como (Italy), 15-16 Sept. 2005. [Cited on page 33.]

T. Raty. Survey on contemporary remote surveillance systems for public safety. *IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 40(5):493–515, Sept. 2010. [Cited on page 32.]

P. Remagnino and S. Velastin. *Intelligent Distributed Video Surveillance Systems*. Institution of Engineering and Technology, 2006. [Cited on page 31.]

M. Rincon, E. Carmona, and M. Bachiller. Segmentation of moving objects with information feedback between description levels. In *Proc. of the Int. Work-conference on the Interplay between Natural and Artificial Computation*, pages 171–181, Murcia (Spain), 18-21 June 2007. [Cited on pages 91 and 122.]

R. Romdhane, F. Bremond, and M. Thonnat. A framework dealing with uncertainty for complex event recognition. In *Proc. of the IEEE Int. Conf. on Advanced Video and Signal based Surveillance*, pages 392–399, Boston (USA), 29 Aug. - 1 Sept. 2010. [Cited on pages 57 and 58.]

Y. Rui, T. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: a power tool for interactive content-based image retrieval. *IEEE Trans on Circuits and Systems for Video Technology,*, 8(5):644 –655, sep 1998. [Cited on page 90.]

M. Ryoo and J. Aggarwal. Semantic representation and recognition of continued and recursive human activities. *Int. Journal on Computer Vision*, 82(1):1–24, Jan. 2009. [Cited on pages 13, 16, 21, 55, 57, and 58.]

M. Saini, M. Kankanhalli, and R. Jain. A flexible surveillance system architecture. In *Proc. of IEEE Int. Conf. on Advanced Video and Signal based Surveillance*, pages 571–576, Genova (Italy), 2-4 Sept. 2009. [Cited on pages 33 and 35.]

P. Salembier and J. Ruiz. On filters by reconstruction for size and motion simplification. In *Proc. of Int. Symposium in Mathematical Morphology*, pages 425–434, Sidney (Australia), Apr. 2002. [Cited on pages 96 and 133.]

J. C. SanMiguel and J. M. Martinez. Robust unattended and stolen object detection by fusing simple algorithms. In *Proc. of IEEE Int. Conf. on Advanced Video and Signal based Surveillance*, pages 18–25, Santa Fe (USA), 1-3 Sept. 2008b. [Cited on pages 48, 54, 70, and 108.]

C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local svm approach. In *Proc. of IEEE Int. Conf. on Pattern Recognition*, volume 3, pages 32–36, Cambridge (UK), 23-26 Aug. 2004. [Cited on page 55.]

C. Shekhar, S. Moisan, R. Vincent, P. Burlina, and R. Chellappa. Knowledge-based control of vision systems. *Image and Vision Computing*, 17(9):667 – 683, july 1999. [Cited on page 90.]

J. Sherrah. Learning to adapt: A method for automatic tuning of algorithm parameters. In *Proc. of Advanced Concepts for Intelligent Vision Systems*, pages 414–425, Sidney (Australia), 13-16 December 2010. [Cited on pages 90 and 97.]

V. Shet, D. Harwood, and L. Davis. Vidmap: video monitoring of activity with prolog. In *Proc. of IEEE Int. Conf. on Advanced Video and Signal-based Surveillance*, pages 224–229, Como (Italy), 15-16 Sept. 2005. [Cited on pages 13 and 57.]

N. Siebel and S. Maybank. The advisor visual surveillance system. In *Proc. of IEEE Eur. Conf. on Computer Vision*, pages 103–111, Prage (Czech Republic), 11-14 May 2004. [Cited on pages 33 and 35.]

C. Simon, J. Meessen, and C. De Vleeschouwer. Using decision trees to recognize visual events. In *Proc. of the 1st ACM workshop on Analysis and retrieval of events/actions and workflows in video streams*, pages 41–48, Vancouver, British Columbia (Canada), 2008. [Cited on pages 174 and 182.]

L. Snidaro, M. Belluz, and G. Foresti. Representing and recognizing complex events in surveillance applications. In *Proc. of IEEE Int. Conf. on Advanced Video and Signal Based Surveillance*, pages 493–498, London (UK), 5-7 Sept. 2007a. [Cited on pages 15 and 16.]

L. Snidaro, G. Foresti, and K. Varshney. Quality-based fusion of multiple video sensors for video surveillance. *IEEE Trans. on System, Man and Cybernetics-Part B. Cybernetics*, 37(4):1044–1051, Aug. 2007b. [Cited on page 121.]

L. Snidaro, M. Belluz, and G. Foresti. Modeling and managing domain context for automatic surveillance systems. In *Proc. of IEEE Int. Conf. on Advanced Video and Signal Based Surveillance*, pages 238–243, Genova (Italy), 2-4 Sept. 2009. [Cited on page 15.]

B. Song, N. Vaswani, and A. Roy-Chowdhury. Closed-loop tracking and change detection in multi-activity sequences. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1–8, San Francisco (USA), 13-18 June 2007. [Cited on page 92.]

G. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *Proc. of IEEE Int. Conf. on Computer Vision*, pages 246–252, Fort Collins (USA), 23-25 June 1999. [Cited on pages 48, 94, 133, 138, 139, 140, and 141.]

R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag, 2011. [Cited on pages 48 and 100.]

Y. Tian, R. Feris, H. Lui, A. Humpapur, and M. Sun. Robust detection of abandoned and removed objects in complex surveillance videos. *IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, (in press) 2010. [Cited on page 71.]

Y.-l. Tian, L. Brown, A. Hampapur, M. Lu, A. Senior, and C.-f. Shu. IBM smart surveillance system (S3): event based video surveillance system with an open and extensible framework. *Machine Vision and Applications*, 19(5):315–327, 2008. [Cited on pages 33 and 35.]

A. Torabi, G. Masse, and G.-A. Bilodeau. Feedback scheme for thermal-visible video registration, sensor fusion, and people tracking. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition Workshops*, pages 15 –22, San Francisco (USA), 13-18 june 2010. [Cited on page 91.]

C. Town. Ontology-driven bayesian networks for dynamic scene understanding. In *Proc. of IEEE Int. Conf. on Computer Vision and Pattern Recognition Workshop*, pages 116–124, Washington DC (USA), 27-02 June 2004. [Cited on page 13.]

C. Town. Ontological inference for image and video analysis. *Machine Vision and Applications*, 17(2): 94–115, Feb. 2006. [Cited on pages 5, 32, 35, 56, 57, and 58.]

K. Toyama and G. Hager. Incremental focus of attention for robust vision-based tracking. *Int. Journal of Computer Vision*, 35(1):45–63, Nov. 1999. [Cited on page 96.]

P. Turaga, R. Chellappa, V. Subrahmanian, and O. Udrea. Machine recognition of human activities: A survey. *IEEE Trans. on Circuits and Systems for Video Technology*, 18(11):1473–1488, Nov. 2008. [Cited on pages 13, 54, 55, and 57.]

A. Tyagi and J. Davis. A context-based tracker switching framework. In *Proc. of IEEE Workshop on Motion and Video Computing*, pages 1–8, Copper Mountain (USA), 8-9 Jan. 2008. [Cited on page 97.]

F. Van der Heijen. Consistency checks for particle filters. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(1):140–145, Jan. 2006. [Cited on pages 124 and 147.]

M. Varela and S. Velastin. Real-time architecture for a large distributed surveillance system. In *Proc. of IEE Intelligent Distributed Systems*, pages 41–45, London (UK), 23 Feb. 2004. [Cited on page 33.]

N. Vaswani. Additive change detection in nonlinear systems with unknown change parameters. *IEEE Trans. on Signal Processing*, 55(3):859–872, Mar. 2007. [Cited on pages 97, 124, 129, 142, 143, 144, 145, 147, 148, 158, 160, 161, 162, 164, 166, and 167.]

P. L. Venetianer, Z. Zhang, W. Yin, and A. J. Lipton. Stationary target detection using objectvideo surveillance system. In *Proc. of IEEE Int. Conf. on Advanced Video and Signal based Surveillance*, pages 242–247, London (UK), 5-7 Sept. 2007. [Cited on pages 33 and 35.]

D. Vernon. Special issue on cognitive vision. *Image and Vision Computing*, 26(1):1–4, Jan. 2008. [Cited on page 4.]

R. Vezzani and R. Cucchiara. Video surveillance online repository (ViSOR): an integrated framework. *Multimedia Tools and Applications*, 50(2):359–380, Feb. 2010. [Cited on pages 15 and 16.]

P. Villegas and X. Marichal. Perceptually-weighted evaluation criteria for segmentation masks in video sequences. *IEEE Trans. on Image Processing*, 13(8):1092–1103, Aug. 2004. [Cited on page 121.]

T. Vu, F. Bremond, and M. Thonnat. Automatic video interpretation: A recognition algorithm for temporal scenarios based on pre-compiled scenario models. In *Proc. of Int. Conf. on Computer Vision Systems*, pages 523–533, Graz (Austria), 1-3 Apr. 2003. [Cited on page 14.]

J. Wang, G. Bebis, M. Nicolescu, M. Nicolescu, and R. Miller. Improving target detection by coupling it with tracking. *Machine Vision and Applications*, 20(4):205–223, June 2009. [Cited on page 91.]

J. Weszka and A. Rosenfeld. Threshold evaluation techniques. *IEEE Trans. on Systems, Man and Cybernetics*, 8(8):622–629, Aug. 1978. [Cited on page 121.]

G. Willems, T. Tuytelaars, and L. Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *Proc. of the IEEE Eur. Conf. on Computer Vision*, volume 2, pages 650–663, Marseille (France), 2008. [Cited on page 55.]

H. Wu and Q. Zheng. Self-evaluation of visual tracking systems. In *Proc. of Army Science Conf.*, Orlando (FL, USA), 29 Nov.-2 Dec. 2004. [Cited on pages 123, 124, 127, 142, 143, 144, and 145.]

H. Wu, A. Sankaranarayanan, and R. Chellappa. Online empirical evaluation of tracking algorithms. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 32(8):1443–1458, Aug. 2010. [Cited on pages 123, 147, 148, 154, 156, 158, 160, 161, 162, 164, 166, and 167.]

H. Zhang, J. Fritts, and S. Goldman. Image segmentation evaluation: a survey of unsupervised methods. *Computer Vision and Image Understanding*, 110(2):186–200, May 2008. [Cited on pages 120, 121, 174, and 182.]

Q. Zhou, L. Ma, and D. Chelberg. Adaptive object detection and recognition based on a feedback strategy. *Image and Vision Computing*, 24(1):80 – 93, Jan. 2006. [Cited on page 91.]